En esta tarea, utilizará las implementaciones de pila y cola de la API Java Collections para crear un programa que determine si una página HTML está bien formateada.

Al completar esta tarea, usted:

- Se familiarizará con los métodos de las clases java.util.Stack y interfaz java.util.Queue

- Trabajará con un tipo de datos abstracto (en concreto, colas) utilizando sólo la interfaz de una implementación. interfaz de una implementación

- Aplicará lo aprendido sobre el funcionamiento de pilas y colas.

**Nota sobre depuración/errores:**

Si se encuentra con errores/fallos/no entiende la salida que el codigo le está dando, por favor postee en el foro de discusión y un asistente técnico le ayudará. NO envíe correos electrónicos a Codio, ya que no revisarán los errores que obtenga.

**Antecedentes**

Las páginas web se escriben en lenguaje de marcado de hipertexto (HTML). Un archivo HTML está compuesto por texto rodeado de etiquetas, que "marcan" el texto especificando su formato, disposición u otra información. Las etiquetas también pueden anidarse.

He aquí un ejemplo sencillo, con las etiquetas resaltadas en negrita:

**<html>**

**<head><title>**Sample HTML page**</title></head>**

**<body>**

This is some **<b>**HTML text**!</b>**

**</body>**

**</html>**

El significado exacto de las etiquetas no es importante (aunque si quieres saber más, ¡puedes apuntarte a nuestro curso "Programación para la Web con JavaScript" como parte de esta serie edX!) pero etiquetas como <body> y <b> se conocen como "etiquetas abiertas" porque indican el inicio de algún formato, y etiquetas como </body> y (con la barra inclinada delante de la palabra) se conocen como etiquetas "cerradas" porque indican el final del formato.

En teoría (aunque no a menudo en la práctica), un HTML bien formateado requiere que las etiquetas estén "equilibradas", es decir, que las etiquetas abiertas vayan acompañadas de su correspondiente etiqueta cerrada en el orden correcto.

Por ejemplo, si ignoramos los espacios en blanco y el texto entre las etiquetas, obtendremos lo siguiente esto:

**<html><head><title></title></head><body><b></b></body></html>**

Tenga en cuenta que hay cierta simetría en las etiquetas HTML, en el sentido de que siempre que cerramos una etiqueta, coincide con la etiqueta abierta más reciente (no cerrada).

Por ejemplo, si resaltamos las etiquetas "title", veremos que una etiqueta cerrada coincide con con la última etiqueta abierta:

<html><head>**<title></title>**</head><body><b></b></body></html>

Y en este caso, la etiqueta "body" cerrada coincide con la etiqueta "body" abierta, que es la última etiqueta abierta que aún no se ha cerrado (ya que la etiqueta "b ya está cerrada):

<html><head><title></title></head>**<body>**<b></b>**</body>**</html>

Algunas etiquetas HTML se cierran solas y no necesitan una etiqueta de cierre.

Por ejemplo, aquí la etiqueta "br" se cierra sola:

<html><head>head><body><b>**<br/>**</b></body></html>

Una etiqueta de autocierre es la que termina con el carácter de barra oblicua, a diferencia de una etiqueta de cierre, que empieza con una.

Es fácil cometer errores en el código HTML. Lo más habitual es olvidarse de cerrar las etiquetas o cerrar las etiquetas anidadas en el orden incorrecto, por ejemplo, algo como esto:

<html><head><title></title><body><b></body></b></html>

En este caso, no hay ninguna etiqueta "head" cerrada, y la etiqueta "body" está cerrada en el orden incorrecto: debería ir después de la etiqueta "b" cerrada.

En esta tarea, escribirás un método que determine si un archivo HTML está formateado correctamente utilizando una pila. Cada vez que tu código encuentre una etiqueta de apertura, deberás colocarla en la pila; cuando encuentres una etiqueta de cierre, deberás colocar la etiqueta en la parte superior de la pila, y si no coinciden, sabrás que el archivo no está formateado correctamente. Más ejemplos y explicaciones a continuación.

**Primeros pasos**

Descargue htmlreader.java y htmltag.java, que contienen código que puede utilizar en esta tarea. No debe cambiar ninguna de estas implementaciones para esta tarea.

HtmlTag.java representa información sobre una única etiqueta HTML. Métodos que pueden serte útiles:

* getElement() Obtiene el nombre del elemento (String) especificado en esta etiqueta.
* isOpenTag() Comprueba si se trata de la etiqueta de apertura. Si la etiqueta es de cierre o de autocierre (por ejemplo, <br/> es una etiqueta de salto de línea que no necesita texto que la acompañe), isOpenTag devolverá false.
* isSelfClosing() Comprueba si una etiqueta es de cierre automático (por ejemplo, <br/>).
* **matches(HtmlTag other)** Comprueba si una etiqueta HtmlTag es la misma etiqueta de apertura/cierre que ella (por ejemplo, <b> y </b> o viceversa).

En HtmlReader.java encontrará un método llamado getTagsFromFile que lee la ruta a un archivo HTML y lo separa en tokens. El resultado es una representación del archivo HTML como una cola de HtmlTags en el orden en que se encontraron. Si lo desea, puede editar este código, pero no debe modifique HtmlTag.java.

Descargue también htmlvalidator.java, que contiene el método no implementado para el código que escribirá en esta tarea.

**Actividad**

En HtmlValidator.java, implemente el método isValidHtml. isValidHtml debe tomar como entrada una Cola de HtmlTags y devolver una Pila de HtmlTags que verifique la correcta estructura de la etiqueta, de acuerdo con la especificación descrita a continuación.

El método debería implementarse como sigue:

Si el archivo HTML está bien formateado, el método debe devolver una Pila vacía.

Por ejemplo:

<html><body><h1>heading</h1><p>paragraph</p></body></html>

En este caso, las etiquetas de cierre coinciden con las de apertura, por lo que el HTML es válido.

Cuando se llega al final del archivo/cola, la pila está vacía.

Si el archivo HTML no está bien formateado, el método debería devolver la Pila en su estado actual (es decir, con sus valores actuales) en el momento en que el código determinó que las etiquetas no estaban equilibradas.

He aquí algunos casos de ejemplo a tener en cuenta:

*Ejemplo nº 1: Etiquetas cerradas en orden incorrecto*

<html><body><p><b>Sentence here**</p>**</b></body></html>

En este caso, usted empujaría todas las etiquetas de apertura en la pila para que se vea así:

<b>

<p>

<body>

y, al encontrar una etiqueta de cierre en la cola, querrá comprobar esa pila para ver si existe la coincidencia correcta. La primera etiqueta de cierre que encuentra es </p>; sin embargo, la última etiqueta de apertura (en la parte superior de la Pila) es <b>.

Eso es malo. Tan pronto como determine que el archivo HTML no es válido, devuelva la pila de etiquetas de apertura sin quitar la etiqueta de apertura que no coincide.

En este caso, la salida esperada sería una Pila que contenga (yendo de abajo hacia arriba): <html><body><p><b>

*Ejemplo nº 2: Etiqueta de cierre sin etiqueta de apertura*

<html><body>Correct<br/><b>Sentence</b>here**</div>**</body></html>

En este caso, la primera etiqueta de cierre que se encuentra (</b>) sí coincide con su etiqueta de apertura, pero la siguiente (</div>) no, por lo que la salida esperada sería una Pila conteniendo (yendo de abajo a arriba): <html><body>

¡Tenga en cuenta que la etiqueta <br/> se cierra sola y no debe colocarse en la Pila!

*Ejemplo nº 3: La etiqueta de apertura nunca se cierra*

**<html><body><b>** Esto es algo de texto En este caso, el método llega al final del archivo/cola y todavía hay elementos en la pila, ya que esas etiquetas de apertura nunca se cerraron. La salida

sería una Pila conteniendo (de abajo hacia arriba): **<html><body><b>**

*Ejemplo nº 4 (¡la parte difícil!): Etiqueta de cierre sin etiqueta de apertura, todo bien hasta entonces*

<html><body><p>Hello, world!</p></body></html>**</p>**

Esto es similar al Ejemplo #2 excepto que ahora cuando se encuentra la etiqueta de cierre que no tiene etiqueta de apertura, la Pila está vacía ya que todo lo anterior se ha comparado. Sin embargo, devolver una Pila vacía significa que el archivo está bien bien formateado. En este caso, sin embargo, necesita devolver null para indicar que el fichero no está bien formateado. Piense en la diferencia entre cuándo devolver null y cuándo devolver una Pila vacía.

Por favor, no cambie la firma del método isValid (su lista de parámetros, nombre y tipo de valor de retorno). Además, no cree ningún archivo .java adicional para su solución y no modifique HtmlTag.java. para su solución y no modifique HtmlTag.java. Si necesita clases adicionales, puede definirlas en HtmlValidator.java. Por último, asegúrese de que su clase HtmlValidator está en el paquete por defecto, es decir, no hay ninguna declaración de "paquete" en la parte superior del código fuente.

**Consejos útiles**

La documentación sobre los métodos de la clase Stack y la interfaz Queue en la última versión de Java está disponible en

• https://docs.oracle.com/javase/8/docs/api/java/util/Stack.html

• https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html

Consulte esta documentación si necesita ayuda para comprender los métodos que tiene a su disposición.

Ten en cuenta que el método HtmlValidator.isValidHtml sólo debe utilizar métodos de la interfaz Queue, aunque la cola se implemente utilizando una LinkedList.

No pasa nada si el método isValidHtml modifica el contenido de la cola que se pasa como entrada. como entrada, por ejemplo, eliminando elementos.

**Antes de enviar**

Asegúrese de que

- su clase HtmlValidator está en el paquete por defecto, es decir, no hay ninguna declaración declaración "package" en la parte superior del código fuente

- su clase HtmlValidator compila y no ha cambiado la firma del método isValidHtml

- no ha creado ningún archivo .java adicional y no ha realizado ningún cambio en HtmlTag.java (no es necesario que envíe este archivo o HtmlReader.java)

1. Descarga la distribución de JUnit en junit-dist.jar Sigue estos pasos para añadir la librería.

2. https://intellij-support.jetbrains.com/hc/enus/

community/posts/360009909039-How-do-I-add-a-build-Path-to-aclass-

folder-

3. Descarga las pruebas en homework2-tests.jar y añádelas a la ruta de compilación del proyecto Eclipse del proyecto Eclipse.

4. Descomprime este archivo en tu ordenador y copia los siete archivos .html en tu proyecto IntelliJ; deberías poder arrastrarlos y soltarlos directamente en IntelliJ. Asegúrate de colocarlos en el directorio raíz de tu proyecto, como hiciste con los dos archivos .jar.

5. Ahora ejecute las pruebas haciendo clic con el botón derecho en homework2-tests.jar en IntelliJ para obtener el menú emergente/contexto y seleccionando "Ejecutar como -->" y luego "Aplicación Java". Application". Deberías ver las pruebas ejecutándose en la consola y debería debería decirte tu puntuación para esta tarea, o "¡Gran trabajo! es del 100%.

Alternativamente, si desea ejecutar el autograder desde la línea de comandos, poner los dos archivos .jar y sus archivos .class para esta tarea en el mismo junto con los archivos .html que descargaste, y ejecútalos:

Mac/Linux: java -cp .:junit-dist.jar:homework2-tests.jar Homework2Grader

Windows: java -cp .;junit-dist.jar;homework2-tests.jar Homework2Grader

Esto añadirá junit-dist.jar y homework2-tests.jar al classpath y luego Java con Homework2Grader como clase principal.