

# HTMLRender.java

**Objective:** To use prior work, logic, string manipulation, and a GUI to render a simple HTML file.

**Background:**

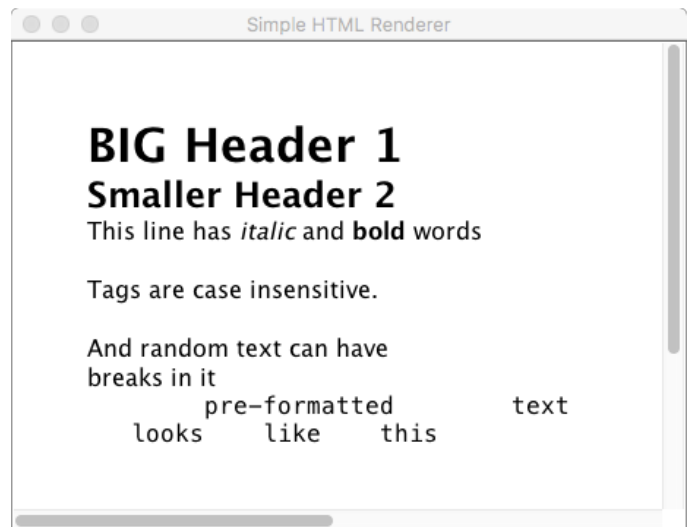
In the prior project, you created a simple HTML tokenizer utility. It reads one line of HTML at a time and returns an array of tokens. The tokens are either contiguous text, punctuation, or HTML tags. In this project, the tokens will be used to render the text on a browser window emulator. The emulator code will be provided as well as documentation on how to use the emulator.

Your job will be to write **HTMLRender** to convert HTML and render an image of the text in a window. GUI software will be provided to you called **SimpleHtmlRenderer**. The process is two steps. First, **HTMLRender** will read an entire HTML file and tokenize it using your `tokenizeHTMLString()` method from **HTMLUtilities**. **HTMLRender** should store all the tokens in one long **String** array or **ArrayList**. Second, **HTMLRender** reads the array, token-by-token and performs the appropriate browser emulator method calls to render the image.

Here is a sample input HTML file:

```
<html><body>
<h1>BIG Header 1</h1>
<h2>Smaller Header 2</h2>
<p>This line has <i>italic</i>
and <b>bold</b> words</p>
<P>Tags are case
insensitive.</P>
And random text can have<br> breaks
in it
<pre>
    pre-formatted      text
  looks   like   this
</pre>
</body></html>
```

and here is the rendered output image.



**Discussion:**

There are a few details to implement in this project.

1. Rendering punctuation is special. Most punctuation does not have a space before it. For example, take the sentence "This is a blast!". `tokenizeHTMLString` returns "This", "is", "a", "blast", and "!". When you print it to the browser, there should be no space between "blast" and "!". This is also true for question marks "?" and periods ".".
2. Tags are case insensitive. The tag "<p>" and the tag "<P>" are treated the same.
3. Plain text should never exceed 80 characters per line. As you render the lines of text, you must count the characters, and remember spaces are characters in the count. Bold and italic text must also not exceed 80 characters per line. The heading formats (<h1> through <h6>) each have different maximum line lengths (see table under **SimpleHtmlRenderer** on a following page). Pre-formatted text is the exception because it must appear as it does in the code regardless of the length.

4. HTMLRender must be able to handle nested tags. The following tags will be presented with the following nesting.

Tags	Name	Nested tags
<html>, </html> <body>, </body>	begin HTML HTML body	all other tags
<p>, </p>	paragraph tags	<b>, </b>; <i>, </i>;  , <hr>
<q>, </q>	quotation tags	 ; <hr>
<b>, </b>	bold tags	 ; <hr>
<i>, </i>	italic tags	 ; <hr>
<hr>	horizontal rule	no nested tags
<h1>, </h1> <h2>, </h2> ... <h6>, </h6>	heading tags	no nested tags
<pre>, </pre>	preformatted text tags	no nested tags

### SimpleHtmlRenderer:

The GUI we will use for this project is called **SimpleHtmlRenderer**, a simple rendering engine that uses print statements to render formatted text to a window. Your **HTMLRender** program will use the following fields to set up the rendering engine:

```
// SimpleHtmlRenderer fields
private SimpleHtmlRenderer render;
private HtmlPrinter browser;
```

Your constructor will contain the following lines:

```
// Initialize Simple Browser
render = new SimpleHtmlRenderer();
browser = render.getHtmlPrinter();
```

Given the code above, all rendering print statements will be prefixed with `browser` (e.g. `browser.print("Hello")`). Below is a list of tags you must render and the corresponding **HtmlRender** print statements.

HTML code	HtmlRender print statement	max line length
text with no tags	<code>browser.print("Hello world")</code>	80
newline	<code>browser.println()</code>	
<p> </p>	same as text with no tags except with blank line after </p>	
<b>Hello</b>	<code>browser.printBold("Hello")</code>	80
<i>Booo</i>	<code>browser.printItalic("Booo")</code>	80
<q> </q>	<code>browser.print("""")</code>	80
<h1>H1</h1>	<code>browser.printHeading1("H1")</code>	40
<h2>H2</h2>	<code>browser.printHeading2("H2")</code>	50

HTML code	HtMLRender print statement	max line length
<h3>H3</h3>	browser.printHeading3("H3")	60
<h4>H4</h4>	browser.printHeading4("H4")	80
<h5>H5</h5>	browser.printHeading5("H5")	100
<h6>H6</h6>	browser.printHeading6("H6")	120
<hr>	browser.printHorizontalRule()	
 	browser.printBreak()	
<pre>Text</pre>	browser.printPreformattedText("Text")	no max

### Assignment:

1. Download and unzip the starter code. It contains the rendering software **SimpleHtmlRenderer.jar** and a starter **HTMLRender.java** file. It also includes sample code **example7.html** for rendering all the tags in this project. You will need to copy over **HTMLUtilities.java** and **FileUtils.java** to perform all the functions required by this project. The base **HTMLRender** code provided has a `run()` method that gives you a sample of printing different tags. To compile and run the sample code, use the following commands:

```
javac -cp .:SimpleHtmlRenderer.jar *.java
java -cp .:SimpleHtmlRenderer.jar HTMLRender
```

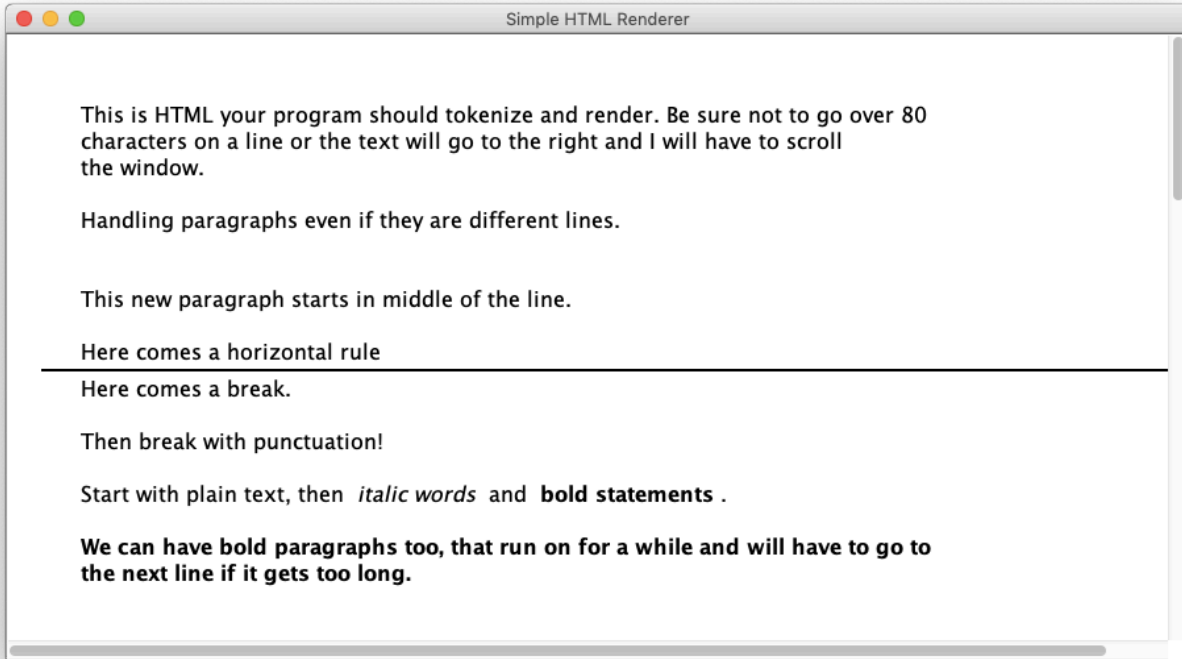
2. Write methods to open the HTML file, read and tokenize each line, and build your array of tokens. Use **HTMLTester** as a model to open the HTML file, read the file, and tokenize each line. It contains code for retrieving the file name from the command line. The input file name should be included on the command line just like **HTMLTester**. To run **example7.html** the command should be:

```
java -cp .:SimpleHtmlRenderer.jar HTMLRender example7.html
```

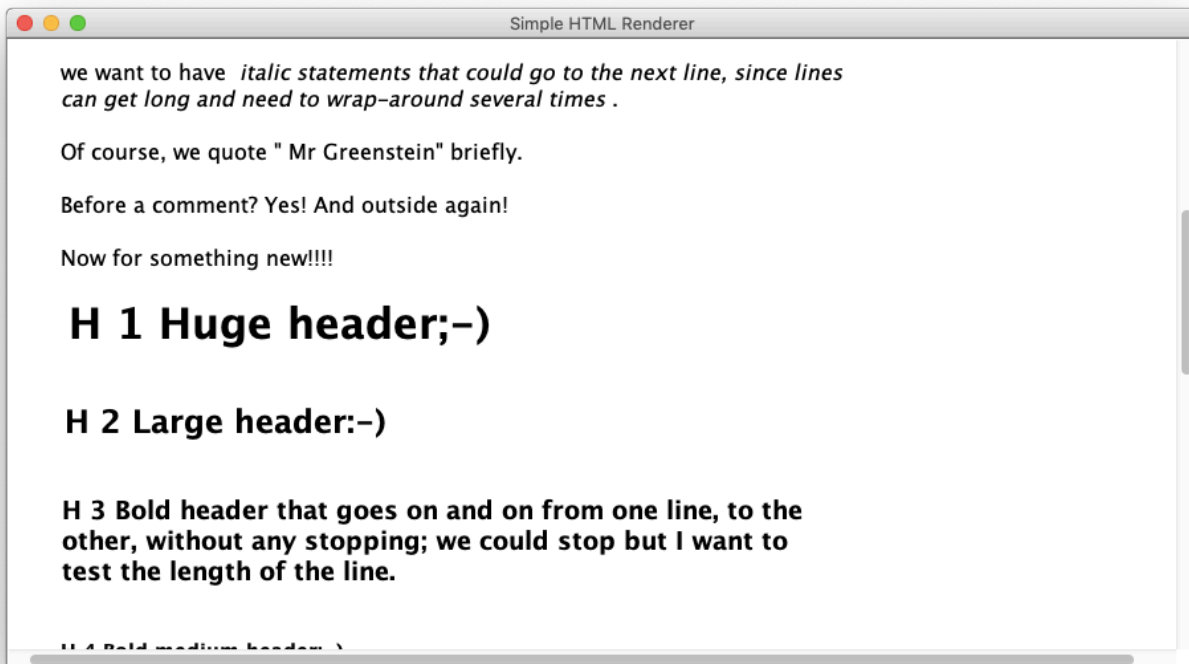
Do not hardcode in "example7.html" or any other filename; otherwise, you will lose points.

3. Write a method to process the tokens in the array and render them onto a window using **SimpleHtmlRenderer**. Your method should use a `while` loop (like `tokenizeHTMLString`) and process the array token-by-token.
4. Use **example7.html** to test your code. A series of sample outputs starts on the next page.

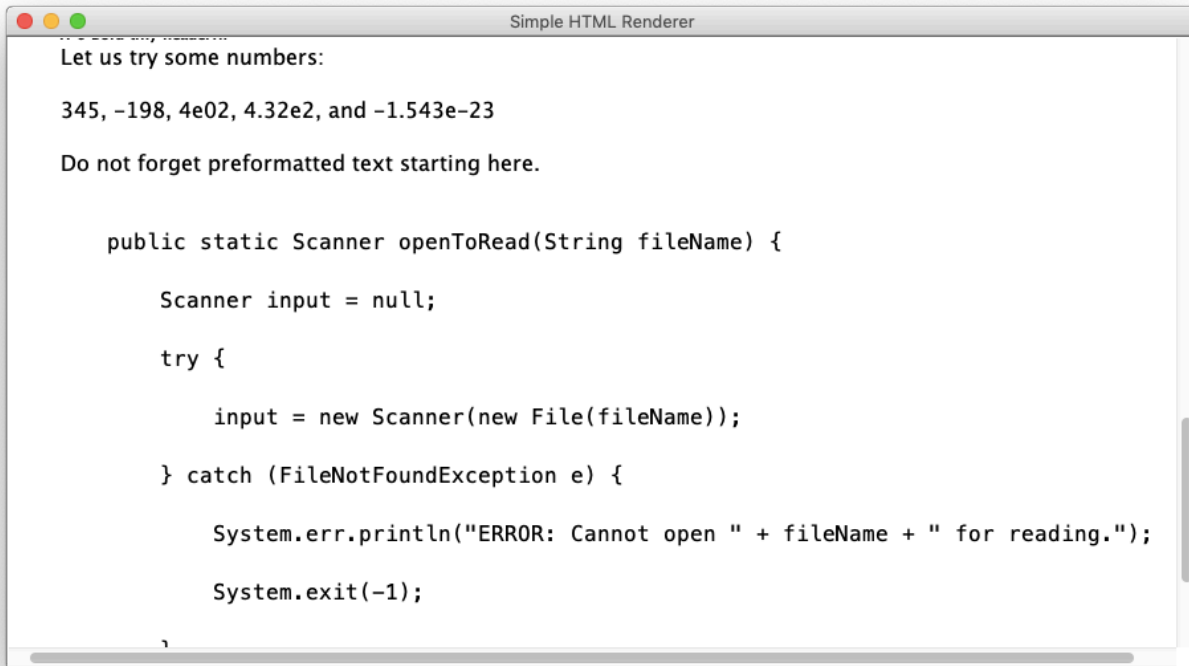
The following graphic shows the rendering of plain text sentences (wrapped-around and not exceeding 80 characters), paragraph spacing, a horizontal rule, line breaks, bold and italic text.



The following graphic shows quoted text, headings H1 through H3, H3 text does not exceed 60 characters per line.



The following graphic shows numbers and preformatted text.



The screenshot shows a browser window with the title "Simple HTML Renderer". The content is preformatted text and code. The text is as follows:

```
Let us try some numbers:  
345, -198, 4e02, 4.32e2, and -1.543e-23  
Do not forget preformatted text starting here.  
  
public static Scanner openToRead(String fileName) {  
    Scanner input = null;  
    try {  
        input = new Scanner(new File(fileName));  
    } catch (FileNotFoundException e) {  
        System.err.println("ERROR: Cannot open " + fileName + " for reading.");  
        System.exit(-1);  
    }  
}
```