# HTMLUtilities.java - Part 2

**Objective**: To use file IO and string manipulation to further enhance a simple HTML tokenizer.

**Background**:
In Part 1 you created **HTMLUtilities** for tokenizing HTML tags, words, punctuation, and numbers. In Part 2 you will enhance your program to handle multi-line constructs, specifically HTML comments and preformatted text.

HTML comments are surrounded by the tags "<!—" and "—>" and treated as a block of code. They can surround anything, including other tags. The purpose of the comment block is to prevent rendering; therefore, comment tags and their contents are not tokenized. A comment block can appear on a line by itself, or in the middle of a line of code, or it could extend for several lines. Below are some examples of HTML comments.

```
<!-- Comment on its own line -->
Tokenize me <!-- Comment in middle of line of code —> <br>
By the way <!— Comment that goes
      for   several lines
that includes <p> tags</p> —> tokenize me
```

Another multi-line construct is "preformatted" lines of code. Sometimes the user does not want the HTML interpreter to format the text. Instead, the lines are already formatted in the HTML code with the proper spacing. When these lines are rendered, they are to appear "as-is." The code is called "preformatted" and the lines are embedded surrounded by the tags "<pre>" and "</pre>". For example, the lines of Java code below represent preformatted code.

```
<pre>
public class Hoot
{
      public static void main(String[] args)
      {
            System.out.println("Hoot hoot");
      }
}
</pre>
```

Without these tags, the Java code would be treated like any other text and all of the formatting above would be lost. In contrast to comments, preformatted code *is* tokenized but each line of code with its formatting is treated as a *single token*. The "<pre>" tag and "</pre>" tag will always be on a line of their own.

**Assignment**:
Use the **HTMLUtilities.java** that you completed in Part 1. You will enhance the `tokenizeHTMLString()` method to handle the comment block and preformat cases. Since these are multi-line code and `tokenizeHTMLString` only tokenizes one line at a time, you will need to create a field that tracks when the tokenizer is inside a multi-line construct.

1. Ignore comments - Comment blocks are ignored by the tokenizer and no tokens are created. Comment tags are special because they look different than other tags. The beginning comment tag is "<!--" and the ending tag is "-->". The tags and whatever those tags surround are ignored by the tokenizer. Comments can sometimes be tricky since they start anywhere. For example:

   ```
   Before a comment <!--Here is a comment
   on several lines
   that ends here.--> And outside again
   ```

   In this example, there are three tokens on the first line and three on the third line, but the other portions are ignored by the tokenizer. When these lines are run through **HTMLTester.java**, it produces the following output:

   ```
   Before a comment <!--Here is a comment
     [token 0]: Before [token 1]: a [token 2]: comment
   ```

```
on several lines

that ends here.--> And outside again
  [token 0]: And [token 1]: outside [token 2]: again
```

The tokenizer always works one line at a time, but comments can span multiple lines. The tokenizer must somehow remember whether it is inside or outside a comment block while it is tokenizing. This can be done by using a field in **HTMLUtilities** that stores the *state* of the tokenization. You will use `enum` to help by inserting the following code into the top of your **HTMLUtilities** program:

```
// NONE = not nested in a block, COMMENT = inside a comment block
// PREFORMAT = inside a pre-format block
private enum TokenState { NONE, COMMENT, PREFORMAT };

// the current tokenizer state
private TokenState state;
```

HTML comments can include nested tags, but these nested text and tags are *not tokenized*. For example, nothing in the following HTML is tokenized.

```
<!-- this has paragraph tags <p>here</p> and break <br>-->
```

To prevent confusion and to simplify our process, we will have the precondition that *comments will not be nested*. You do not need to detect nested comments. (e.g. " `<!--  <!--  -->  -->` ")

Use the input file **example5.html** to test your code with comment blocks.

2. Preformatted text - Each line of preformatted text is treated as one token, including the formatting (spacing and tabs) as it appears in the code. Preformatted text can also span multiple lines; therefore, the tokenizer must somehow remember whether it is inside or outside a preformatted block of code. The field used to store the state of comments can be used since comments and preformatted text are mutually exclusive constructs.

   The tags for preformatted code are "`<pre>`" and "`</pre>`" and each tag will appear on its own line in the code.

   Use the input file **example6.html** to test your code with preformatted text. Below are a few lines produced by **HTMLTester** and the **example6.html** file.

```
<pre>

  [token 0]: <pre>

   public static Scanner openToRead(String fileName) {

  [token 0]:     public static Scanner openToRead(String fileName) {

          Scanner input = null;

  [token 0]:            Scanner input = null;
```