The *ColorizeFSM* project

This is a pair programming assignment that you must do with your assigned partner. You must use pair programming techniques. That is, you cannot simply divide the work and program independently. Developing the ability to create software in pairs is vital to your success, both in this course and professionally.

I suggest that you quickly read this entire document once to get an overview, then read it again slowly a couple of times to get the details.

In this project, you will write a program that takes a legal Java program (in a single file) as input and produces an HTML file that "colorizes" the following elements of the program:

- string literals (red),
- character literals (magenta),
- single-line comments (green),
- multi-line comments (cyan),
- keywords (blue) and
- non-keyword identifiers (orange).

You will do ONLY the Finite State Machine (FSM) that processes the input file, character by character; we have written all the other supporting code, including methods that write HTML.

For example, suppose the input file is as shown in the first picture to the right. Then the correct HTML output is shown in the picture on the NEXT page. (Note that three long lines wrap in that picture). The Eclipse browser renders that HTML as shown in the second picture to the right.

It is also important that you write your program in such a way that no possible input can cause the program to get into an infinite loop.

Proceed as follows:

```
/*
 * Multiline comment.
 * Yep.
 */
public class __HelloWorldTest {
    public static void main( String args[] )  {
        // Testing, 1, 2, 3.
        System.out.println( "Hello," + " \"World\"!" );
        System.out.println('?');
    }
}
```

```
/*
 * Multiline comment.
 * Yep.
 */
public class __HelloWorldTest {
    public static void main( String args[] )  {
        // Testing, 1, 2, 3.
        System.out.println( "Hello," + " \"World\"!" );
        System.out.println('?');
    }
}
```

1. ***Begin*** drawing a FSM that represents how to process the Java file. Each Event is the next character in the Java file. ***Just do 2 or 3 states for now.***

   For example, do the *Start* state and the edges out of it: a double quote (which leads to the *String Literal* state), a slash (which leads to a *Maybe Comment* state), etc. Then do one of the states you just drew. Stop after 2 or 3 states.

2. Now complete this partial diagram for the ColorizeFSM (that's a PDF, here is a Word version), by labeled the arcs not yet labeled and filling in empty boxes with appropriate actions.

3. ***After*** you do the partial diagram yourself, check out the ***ColorizeFSM*** project if you have not already done so. Example the ***FSM.pdf*** file and the FSM diagram in it. It should be similar to the one you did.

   While this is certainly not the only possible correct FSM for doing the required colorizing, it is the FSM that you will implement.

4. Examine the ***Colorize*** class and the ***FiniteStateMachine***

```
<html>
<style type="text/css">
    .slcomment { color: green; }
    .mlcomment { color: cyan; }
    .string { color: red; }
    .character { color: magenta; }
    .keyword { color: blue; }
    .identifier { color: orange; }
</style>
<body>
<pre>
<span class="mlcomment">/*
 * Multiline comment.
 * Yep.
 */</span>
<span class="keyword">public</span> <span class="keyword">class</span> <span class="identifier">__HelloWorldTest</span> {
    <span class="keyword">public</span> <span class="keyword">static</span> <span class="keyword">void</span> <span
class="identifier">main</span>( <span class="identifier">String</span> <span class="identifier">args</span>[] )  {
       <span class="slcomment">// Testing, 1, 2, 3.</span>
        <span class="identifier">System</span>.<span class="identifier">out</span>.<span class="identifier">println</span>( <span
class="string">"Hello,"</span> + <span class="string">" \"World\"!"</span> );
        <span class="identifier">System</span>.<span class="identifier">out</span>.<span class="identifier">println</span>(<span
class="character">'?'</span>);
    }
}

</pre></body></html>
```

class. ***You will do all your work in the latter.***

In doing your work, you will use the following methods from the ***HTMLColorWriter*** class:

```
beginColor     endColor    write
```

Use only those methods from that class; no further understanding of HTML is required. These methods use CSS (Cascading Style Sheets) to color things rather than using the HTML font tag (this is the modern way to do formatting in web documents).

5. We supplied many test cases. Ask your instructor to show you how to use them.

6. Implement and test ***FiniteStateMachine***. To test it for a particular input file, run the **Colorize.java** program as a Java Application. You can either change the code to use a different input file, or enter the file location as a command-line argument. To test it for *all* of our test files (you should certainly do this before your final submission), run **ColorizeTest.java** as a JUnit Test.