CSCI 3353 Object Oriented Design
Homework Assignment 8
Due Friday, November 18

In this assignment you will use the State pattern to read and interpret CSV files. CSV stands for "Comma Separated Values". A CSV file is a text file that can be interpreted as a table of values. Each line of the file denotes a row of the table, and the values in each row are separated by commas. For example, the following line denotes three values:

```
value one,abcd,a longer third value
```

You can also place a value inside of double quotation marks. These quotation marks can only be at the beginning and end of the field, and are not part of the value. For example, the following line has three values. The first two are identical to the line above, but the third value is illegal because the quotes are in the middle of the field:

```
"value one","abcd",a longer "third" value
```

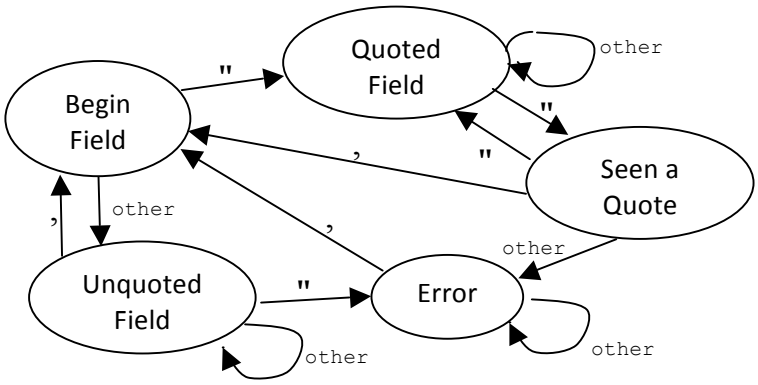
Quoted values are useful when you want the value to contain a comma. (Without quotes, the comma would be interpreted as a field delimiter.) But how do you get a double-quote mark into a value? The answer is to double it. For example, consider the following line:

```
"a simple, ""better"" value","""absolutely""",but ""not"" good
```

This line contains three values: The first is: *a simple, "better" value*. The second is: *"absolutely"*. The third is an error, because quotes are not allowed in the middle of an unquoted field.

Download the file *text.csv*, and make sure you are able to determine its values. Note that it contains one error value. Also note that the lines don't have the same number of values.

The rules for extracting values from a line of text can be expressed by the following state diagram:



Note that when the machine is in the Error state, it ignores all characters until it hits a comma. This is a rudimentary form of error recovery. Let's assume that the value that the machine assigns to an error field is always "ERROR".

Now we can talk about what you have to do.

Use the State pattern to implement this state diagram. You should have a class for each state, as well as a class for the "CSV state machine".

The CSV class is responsible for keeping track of the output values for a single line of input text. It should store the output as a list of strings, where each string is one of the output values. The state classes are responsible for determining how each input character affects the output. For example, when the "Unquoted Field" state sees an "other" character, it knows that this character belongs to the current output field. When the "Unquoted Field", "Error", or "Seen a Quote" states see a comma, they know that the current output field is complete. These states communicate this information to the CSV class (via appropriate methods), so that it can maintain its list of output strings. The CSV class should also have a public method that returns this list to a client.

Please download my client file *HW8Test.java*. Examining it should help clarify how a CSV class is used. This file reads from a file named "text.csv". This is the file I will use to test your code. While you are developing (and debugging) your code, you might want to hardcode a particular input string. Just change the line

```
Scanner sc = new Scanner(new FileInputStream("text.csv"));
```

to something like this:

```
Scanner sc = new Scanner("v1,\"v \"\"2\"\", perhaps\",v3");
```

WHAT TO SUBMIT: Please strip all java files of any package declarations and zip them into a single file. Then submit that file to Canvas.