**CSC236 Data Structures in Java  
Lab 2**

In this lab you will you will write a program to validate html files.

In html, tags are element names enclosed in angle brackets < >. Most tags have start and end tags which enclose some document element(s) and possibly other tags. End tags have the same name as the corresponding start tag, preceded by a '/':

<p>Arrays are <b>objects</b>, which means that when we declare an array, the <b><i>name</i></b> of the array is a <b>reference</b>

<a hef="array.jpg"></p>.

Start tags can contain attributes which provide additional information about the element, such as <img src="cats.jpg">

Some tags do not surround any content. These "empty elements" have start tags but no end tags.

**Restrictions for This Validator**

There are multiple versions of html with different rules. We are going to simplify a bit for the purposes of this lab. We are going to use the following rules for valid html:

* tags are case insensitive
* the following tags are empty, and so have no corresponding close tag: <area>, <base>, <br>, <col>, <command>, <embed>, <hr>, <image>,<input>, <keygen>, <link>, <meta>, <param>, <source>, <track>, <wbr>
* all other tags must have a close tag
* tags must be nested correctly, so <p><b> . . .</b></p> is valid but <p><b> . . .</p></b> is not.

**Input**

The input is an html file which may or may not follow the restrictions described above.

**Output**

* each time a pair of tags is matched, print a message such as: "matched: <p> and </p>"
* each time a tag with no end tag is processed, print a message such as "no match needed: <br>"
* each time a start tag has no matching end tag, print a message such as: "missing end tag: <li>"
* each time there is no start tag to match an end tag, print a message such as "missing start tag: </head>
* if there are no errors in the file, print a message saying that the file is valid; if there are any errors in the file, print a message saying that the file is invalid

**Requirements**

1. Prompt for the name of the file to be validated. Create a Scanner for the file and pass it to your validator.
2. Your validator should use a stack to check for correct nesting of tags. The algorithm is similar to the algorithm for checking for balanced parentheses in our textbook. Open tags are pushed onto the stack. Each close tag is compared with the top of the stack to check for a match.
3. Don't forget that some tags have no close tag.
4. Create class for tags. This class should not have set or get methods. The Tag class should contain the list of tags with no end tag and a method to test whether a tag has no end tag. Your tag class should also have a method to test whether a pair of tags are matching start and end tags. You will probably need some additional methods in the Tag class.
5. You need to extract the tags from the input file. **Attributes are not stacked.** My advice is to write your own "split" method that will take a line of the input file and return an array or ArrayList of the tags (modified to remove attributes) in a line of input. A good place for this method is to make it a static method in the Tag class.