CS61B, Fall 2013 **HW #4** **P. N. Hilfinger**

Due: Fri., 4 October 2013

Use `hw init hw4` to create a working directory to hold your answers. These files are also in the directory `~cs61b/code/hw4`. Use `hw submit` to copy your final solution to a `hw4-`N entry in your tags repository directory.

The purpose of this homework is to help you learn a few things about text input and output and other text manipulation. Please read §9.2 (especially §9.2.1), §9.5, and §9.6 of *A Java Reference* on the class web page. Also, take a look at the Javadoc documentation for the classes `java.util.Scanner`, and `java.util.regex.Pattern`, and `java.util.regex.Matcher`. It's up to you to understand this documentation to the point where you can figure out how to answer the questions below.

The file `Tester.java` contains a number of routines for testing parameters you provide for `Scanner` methods. First, read and understand these methods. Feel free to adapt these to your own use; any changes you make to them will be thrown away before testing. Only the constants in `HW4.java` that you supply are significant.

1. You have a file containing a sequence of items separated by commas (where the commas may be followed by arbitrary whitespace). There is no white space at the start of the file. Each item must be one of

1. A signed floating-point or integer numeral (in Java, represented as type `double`);
2. A string that starts with a non-whitespace character other than a digit, minus sign, or decimal point, and that contains characters other than commas (it may include blanks).

In the file `HW4.java`, fill in the definitions of `HW4.DELIM_P1` and `HW4.OKSTRING_P1` with pattern strings that match delimiters and string items as described above, so that `Tester.p1` identifies and prints out the items in an input file. We've provided a test file, `tests/p1-2.inp` but will test with other cases, so make your solution general.

2. Add definitions for `HW4.DOUBLE_P2` and `HW4.ANY_STRING_P2` so that `Tester.p2` performs the same task as `Tester.p1`, using `findWithinHorizon`. Again, `tests/p1-2.inp` is a sample test case, but we will use others.

More problems on next page

3. Fill in a definition for `HW4.HTML_P3` with a pattern string that matches an HTML markup with one of the forms

A. <*TAG ATTRIBUTE*="*VALUE*"/> or

B. <*TAG ATTRIBUTE*="*VALUE*"> or

C. <*TAG*/> or

D. <*TAG*> or

E. </*TAG*>

—where *TAG* and *ATTRIBUTE* are strings of letters and *VALUE* is a sequence of characters not including double quote—so that `Tester.p3()` correctly prints out the tag and, if present, the attribute and value for all such markups in a given input file. Assume that there *may* be extra blanks around the *TAG*, '=', and before the closing '/>' or '>'.

4. Fill in the definition of HW4.FORMAT˙P4 so that Tester.p4() correctly prints a list of Name/Value pairs in the following format:

```
Mercury␣␣|␣␣␣32.98
Venus␣␣␣␣|␣␣␣67.23
Earth␣␣␣␣|␣␣␣92.96
Mars␣␣␣␣␣|␣␣141.64
Jupiter␣␣|␣␣483.63
Saturn␣␣␣|␣␣888.19
Uranus␣␣␣|␣1783.95
Neptune␣␣|␣2798.84
```

That is, the name is left-justified in the first 8-character field; the value is right-justified in a 7-character field and rounded to two decimal places; the fields are separated by a vertical bar surrounded by blanks; and the line is terminated by a newline.