Review/Warm-up Composition, Classes, Methods, Arrays and File Input

50 points - See Canvas for due date

Minimal Documentation Required (name at the top of each file, brief description of each class, document any shortcomings, cite any resources used)

In this assignment you will create

- An Address class
- A Letter class that is composed of two Address objects
- A PostOffice class that is composed of 1 to 1000 Letters

Address

Create an **Address** class that contains the following attributes as **private** instance fields:

- street
- city
- state
- zip code

The class should have the following instance methods:

- an explicit value constructor that is passed a street, city, state, and zip code – all as Strings
- a toString method that returns a String that contains the address info in the following format:

```
Street value<\n>
City value,<space>State value<space><space>Zip code
value<\n>
```

- set and get methods for each field as you deem necessary

Letter

Create a **Letter** class that contains the following attributes as **private** instance fields:

- name of whom letter is to
- Address of whom letter is to
- name of whom letter is from
- Address of whom letter is from
- weight of the letter in ounces

The class should have the following instance behaviors:

- an explicit value constructor that is passed
  - the name of whom the letter is to
  - the street, city, state and zip of whom the letter is to
  - the name of whom the letter is from
  - the street, city, state and zip of whom the letter is from
  - the weight (type double)
- a toString method that returns a String that contains the Letter info in the following format:

```
-----------------------------------------------------------
-----------------
From: From
value                                              Postage
value
From Address value (be sure and use Address toString)

                    To: To value
                    To Address value (be sure and use
Address toString)

-----------------------------------------------------------
-----------------
```

- a compareTo method that is passed a Letter object and compares the current Letter to the one passed in first by zip code, then by street value of the to address if the zip codes are the same. The method should return 0 if the two Letter objects are the same, a negative integer value if the current Letter is less than the Letter passed in, and a positive integer value if the current Letter is greater.
- a **static** 'public static double getPostage(double weight)' method that accepts a weight value and calculates the appropriate postage at .46 cents per ounce or portion thereof.

PostOffice

Create a **PostOffice** class that contains the following attributes as **private** instance fields:

- an array of type Letter. Size will be determined by a read-through of a text file, prior to second read-and-load of the text file.

The class should have the following behaviors:

- a default value constructor that creates the Letters array reference (and only the reference – not an actual instance of the array yet.)
- a readLetters instance method that is passed the name of a file from which to read Letter data. The method should:
    - open the file specified
    - count the number of lines of text data
    - determine how many Letter objects is represented by that many lines of text
    - reposition the file cursor to the beginning of the file (close and reopen or
    - read the Letter data again from the file and create Letter objects
    - assign each Letter object to an element of your array of type Letter
    - close the file when finished
- a sortLetters instance method that sorts the Letters in your array in ascending order. This method should call the compareTo method provided by the Letter class to sort. You may use any sorting routine you wish (see SortSearchUtil.java – a basic utility file - under Course Documents if you don't have one of your own.
- a printLetters instance method. The method should:
    - print the count of Letters followed by the total postage followed by each Letter (make sure you use the toString method provided by the Address and Letter classes for this)
- a main method which is of course static. Here is the code for your main method:

```
PostOffice postOffice = new PostOffice( );

postOffice.readLetters( "letters.in" );
postOffice.sortLetters( );
postOffice.printLetters( );
```

- helper methods as you deem necessary to aid the instance methods

---

Input file (letters.in) specifications

The input file will have at least one entry and at most 1000. Each entry will be formatted as follows:

```
* Name of whom letter is to
* Street address
* City,<space>State<space><space>Zip code (may be 5 digits OR 5
digits followed by '-' followed by 4 digits)
* Name of whom letter from
* Street address
* City,<space>State<space><space>Zip code (may be 5 digits OR 5
digits followed by '-' followed by 4 digits)
* Weight
```

Here is a small sample input file to help clarify:

```
Stu Steiner
123 Slacker Lane
Slackerville, IL  09035
Tom Capaul
999 Computer Nerd Court
Dweebsville, NC  28804-1359
0.50
Tom Capaul
999 Computer Nerd Court
Dweebsville, NC  28804-1359
Chris Peters
123 Some St.
Anytown, CA  92111-0389
1.55
```

Be sure your input file contains data that will properly test all aspects of sorting.

Output sample based on sample input file

```
Letter count: 2
Total postage: $1.38

----------------------------------------------------------------
---------
From: Tom
Capaul                                                    $.46
999 Computer Nerd Court
Dweebsville, NC  28804-1359

                        To: Stu Steiner
                        123 Slacker Lane
                        Slackerville, IL  09035
```

```
-----------------------------------------------------------------
---------

-----------------------------------------------------------------
---------
From: Chris
Peters                                                      $.94
123 Some St.
Anytown, CA 92111-0389
                        To: Tom Capaul
                        999 Computer Nerd Court
                        Dweebsville, NC  28804-1359
-----------------------------------------------------------------
---------
```

To Turn In

Submit a zip file named with your last name, followed by the first initial of your first name, followed by hw1 (e.g: **peterschw1.zip**) that contains your source files and the input file that you used to test your program.

Get started ASAP!