**Programming Project 1**

The first programming project involves writing a program that computes the minimum, the maximum, and the average weight of a collection of weights represented in pounds and ounces that are read from a data file. This program consists of two parts. The first part is the Weight class and the second part is the Program Core.

The Weight Class is specified in integer pounds and ounces stored as a double precision floating point number, that is, a percentage of a pound. It should have five public methods and two private methods:

1. A public ***constructor*** that allows the pounds and ounces to be initialized to the values supplied as parameters.
2. A public instance method named ***lessThan*** that accepts one weight as a parameter and returns whether the weight object on which it is invoked is less than the weight supplied as a parameter.
3. A public instance method named ***addTo*** that accepts one weight as a parameter and adds the weight supplied as a parameter to the weight object on which it is invoked. It should normalize the result.
4. A public instance method named ***divide*** that accepts an integer divisor as a parameter. It should divide the weight object on which the method is invoked by the supplied divisor and normalize the result.
5. A public instance ***toString*** method that returns a string that looks as follows: *x* lbs *y* oz, where *x* is the number of pounds and *y* the number of ounces. The number of ounces should be displayed with four places to the right of the decimal.
6. A private instance method ***toOunces*** that returns the total number of ounces in the weight object on which is was invoked. Two processes occur in this method. The first process takes the value read in from a data file that is a percentage of a pound and it is checked that is one of fifteen acceptable numbers using modulo of integer division (See the Conversion Table). If the value is not an acceptable number, then a message is displayed indicating that the value is not acceptable. That value is not included in the array used to determine the average. If the value is acceptable, the percentage of a pound is converted into one of the fifteen ounces with integer division. That value is returned.
7. A private instance method ***normalize*** that normalizes the weight on which it was invoked by ensuring that the number of ounces is less than the number of ounces in a pound.

Both instance variables must be private. In addition, the class should contain a *private named constant* that defines the number of ounces in a pound, which is 16. The class must not contain any other public methods.

The second part is the Program Core. It should consist of the following four class (static) methods.

1. The ***main*** method that reads in the file of weights and stores them in an array of type Weight. It should then display the smallest, largest, and average weight by calling the remaining three methods. The user should be able to select the data file from a default directory by using the JFileChooser class. Reading the data can follow the coding used in ReadEmails.java found in Week 2 |Code Examples. Attached to this assignment are two files containing weights: *PRJ1WeightsLess* and *PRJ1WeightsMore*. The second file contains more than 25 entries. The data file contains one weight per line divided into two parts: the pound as integer and the percentage of a pound as a float. If the number of weights in the file exceeds 25, an error message should display, and the program should terminate with a message.
2. A private class method named ***findMinimum*** that accepts the array of weights as a parameter together with the number of valid weights it contains. It should return the smallest weight in that array.
3. A private class method named ***findMaximum*** that accepts the array of weights as a parameter together with the number of valid weights it contains. It should return the largest weight in that array.
4. A private class method named ***findAverage*** that accepts the array of weights as a parameter together with the number of valid weights it contains. It should return the average of all the weights in that array.

Be sure to follow good programming style, which means making all instance variables private, naming all constants and avoiding the duplication of code. Furthermore, you must select enough different data files for testing and enough weights to completely test the program.