Lab: Implementing a Custom WritableComparable

In this lab, you will create a custom WritableComparable type that holds two strings.

Test the new type by creating a simple program that reads a list of names (first and last) and counts the number of occurrences of each name.

The mapper should accepts lines in the form:

`lastname firstname` *`other data`* The goal is to count the number of times a lastname/firstname pair occur within the

dataset. For example, for input:

```
Smith Joe 1963-08-12 Poughkeepsie, NY
Smith Joe 1832-01-20 Sacramento, CA
Murphy Alice 2004-06-02 Berlin, MA
```

We want to output:

```
(Smith,Joe)    2
(Murphy,Alice) 1
```

Instructions

1. Input Data

   The `nameyeartestdata` file contains the data for this lab.

2. Source code

   You will find the source files in the `src/stubs` directory. You will also find a `src/hints` directory that contains a copy of the source files modified to make the assignment easier to complete, and a `src/solution` directory that contains a copy of the source files modified to complete the assignment.

3. Complete the `StringPairWritable`, `StringPairMapper`, and `StringPairTestDriver` classes in the `src/stubs` directory to count the number of occurrences of each name.

4. Assemble and test your solution.

StringPairWritable

You need to implement a WritableComparable object that holds the two strings. The stub provides an empty constructor for serialization, a standard constructor that will be given two strings, a `toString()` method, and the generated `hashCode()` and `equals()` methods. You will need to implement the `readFields()`, `write()`, and `compareTo()` methods required by WritableComparables.

Note that Eclipse automatically generated the `hashCode()` and `equals()` methods in the stub file. You can generate these two methods in Eclipse by right-clicking in the source code and choosing 'Source' > 'Generate hashCode() and equals()'.

Name Count Test Job

The test job requires a Reducer that sums the number of occurrences of each key. You may either write a `SumReducer` class or use the `org.apache.hadoop.mapreduce.lib.reduce.LongSumReducer` library class, which does exactly that.

Use the simple test data in `nameyeartestdata` to make sure your new type works as expected.

You may test your code using local job runner or by submitting a Hadoop job to the (pseudo-)cluster as usual. If you submit the job to the cluster, note that you will need to copy your test data to HDFS first.