Implementing a Custom WritableComparable

|  |  |
| --- | --- |
| **Exercise Dir** | ~/workspace/writables |
| **Eclipse Proj** | writables |
| **Java Files** | StringPairWritable.java  StringPairMapper.java  StringPairTestDriver.java |
| **JAR File** | stringpairtest.jar |
| **Data File** | nameyeartestdata |

**In this exercise, 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

NOTE: You will use your custom WritableComparable type in a future exercise, so make sure it is working with the test job now.

**StringPairWritable**

You need to implement a WritableComparable object that holds the two strings. The fixme 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 fixme 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. This is the same function that the SumReducer used previously in wordcount, except that SumReducer expects Text keys, whereas the reducer for this job will get StringPairWritable keys. You may either re-‐write SumReducer to accommodate other types of keys, or you can use the LongSumReducer Hadoop library class, which does exactly the same thing.

You can use the simple test data in ~/materials/data/nameyeartestdata to make sure your new type works as expected.

You may test your code using local job runner or 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.

**END**