Creating an Inverted Index

Eclipse project: `inverted_index`
Java files:
`IndexMapper.java` (Mapper)
`IndexReducer.java` (Reducer)
`InvertedIndex.java` (Driver)
Data files:
`~/materials/data/invertedIndexInput.tgz`
Exercise directory: `~/workspace/inverted_index`
JAR File: `invertedindex.jar`

In this exercise, you will write a MapReduce job that produces an inverted index.

For this lab you will use an alternate input, provided in the file `invertedIndexInput.tgz`. When decompressed, this archive contains a directory of files; each is a Shakespeare play formatted as follows:

0	Hamlet
1	
2	DRAMATIS PERSONAE
3	
4	CLAUDIUS king of Denmark. (KING CLAUDIUS:)
5	
6	
7	HAMLET son to the late, and nephew to the present king.
8	
9	
10	POLONIUS lord chamberlain. (LORD POLONIUS:)

Each line contains:

```
    Line number
    separator: a tabcharacter value:
    the line of text
```

This format can be read directly using the `KeyValueTextInputFormat` class provided in the Hadoop API. This input format presents each line as one record to your Mapper, with the part before the tab character as the key, and the part after the tab as the value.

Given a body of text in this form, your indexer should produce an index of all the words in the text. For each word, the index should have a list of all the locations where the word appears. For example, for the word 'honeysuckle' your output should look like this:

```
honeysuckle    2kinghenryiv@1038,midsummernightsdream@2175,...
```

The index should contain such an entry for every word in the text.

Prepare the Input Data

1. Extract the `invertedIndexInput` directory and upload to HDFS:

```
    $ cd ~/materials/data
    $ tar zxvf invertedIndexInput.tgz
    $ hdfs dfs -put invertedIndexInput invertedIndexInput
```

Define the MapReduce Solution

Remember that for this program you use a special input format to suit the form of your data, so your driver class will include a line like:

```
job.setInputFormatClass(KeyValueTextInputFormat.class);
```

(Don't forget to import this class for your use.)

Retrieving the File Name

Note that the exercise requires you to retrieve the file name – since that is the name of the play. The Context object can be used to retrieve the name of the file like this:

```
    FileSplit fileSplit = (FileSplit) context.getInputSplit();
    Path path = fileSplit.getPath();
```

```
String fileName = path.getName();
```

Build and Test Your Solution

Test against the `invertedIndexInput` data you loaded above.

Hints

You may like to complete this exercise without reading any further, or you may find the following hints about the algorithm helpful.

The Mapper

Your Mapper should take as input a key and a line of words, and emit as intermediate values each word as key, and the key as value.

For example, the line of input from the file 'hamlet':

`282 Have heaven and earth together` produces intermediate output:

Have	*hamlet@282*
Heaven	*hamlet@282*
And	*hamlet@282*
Earth	*hamlet@282*
Together	*hamlet@282*

The Reducer

Your Reducer simply aggregates the values presented to it for the same key, into one value. Use a separator like ',' between the values listed.

END