Writing a MapReduce Java

Program

|  |  |
| --- | --- |
| **Exercise Dir** | ~/workspace/averagewordlength |
| **Eclipse Proj** | averagewordlength |
| **Java Files** | AverageReducer.java (Reducer)  LetterMapper.java (Mapper)  AvgWordLength.java (Driver) |
| **JAR File** | avgwordlength.jar |
| **Test Data** | shakespeare |

**In this exercise you write a MapReduce job that reads any text input and computes the average length of all words that start with each character.**

For any text input, the job should report the average length of words that begin with ‘a’, ‘b’, and so forth. For example, for input:

*“No now is definitely not the time”*

The output would be:

|  |  |
| --- | --- |
| N | 2.0 |
| n | 3.0 |
| d | 10.0 |
| i | 2.0 |
| t | 3.5 |

**The Algorithm**

The algorithm for this program is a simple one-pass MapReduce program:

**The Mapper**

The Mapper receives a line of text for each input value. (Ignore the input key.) For each word in the line, emit the first letter of the word as a key, and the length of the word as a value. For example, for input value:

*No now is definitely not the time*

Your Mapper should emit:

|  |  |
| --- | --- |
| N | 2 |
| n | 3 |
| i | 2 |
| d | 10 |
| n | 3 |
| t | 3 |
| t | 4 |

**The Reducer**

Thanks to the shuffle and sort phase built in to MapReduce, the Reducer receives the keys in sorted order, and all the values for one key are grouped together.

The Mapper output above, the Reducer receives this:

|  |  |
| --- | --- |
| N | (2) |
| d | (10) |
| i | (2) |
| n | (3,3) |
| t | (3,4) |

The Reducer output should be:

|  |  |
| --- | --- |
| N | (2) |
| d | (10) |
| i | (2) |
| n | (3,3) |
| t | (3,4) |

**Write the Program in Java**

We’ve provided fixme files for each of the Java classes for this exercise:

LetterMapper.java (the Mapper)

AverageReducer.java (the Reducer)

AvgWordLength.java (the driver).

1. If you are using Eclipse, open the fixme files (located in the src/fixme package) in the averagewordlength project. If you prefer to work in the shell, the files are in:

$ ~/workspace/averagewordlength/src/fixme

You may wish to refer back to the wordcount example (in the wordcount project in Eclipse or in ~/workspace/wordcount) as a starting point for your Java code. Here are a few details to help you begin your Java programming.

1. Define the driver

This class should configure and submit your basic job. Among the basic steps here, configure the job with the Mapper class and the Reducer class you will write, and the data types of the intermediate and final keys.

1. Define the Mapper

Note these simple string operations in Java:

str.substring(0,1)

// string : first letter of str

str.length()

// int : length of str

1. Define the Reducer:

In a single invocation, the reduce() method receives a string containing one letter (the key) along with an iterable collection of integers (the values), and should emit a single key‐value pair: the letter and the average of the integers.

1. Compile your classes and assemble the jar file

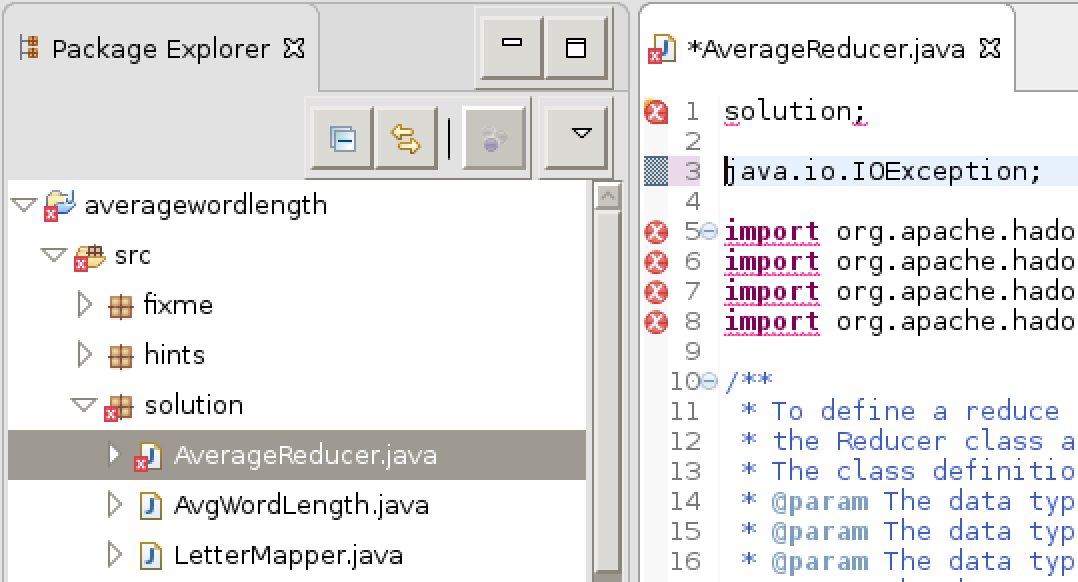
To compile and jar, you may either use the command line javac command. Or follow the steps below (“Using Eclipse to Compile Your Solution”) to use Eclipse.

**Use Eclipse to Compile Your Solution**

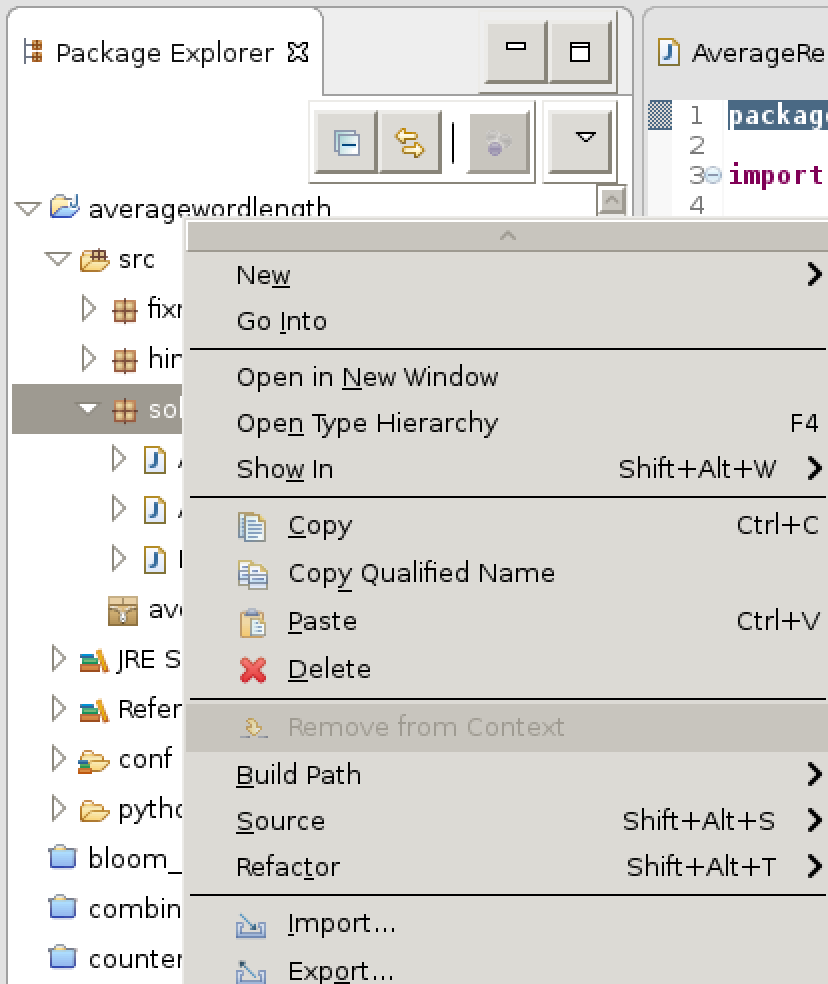
Follow these steps to use Eclipse to complete this exercise.

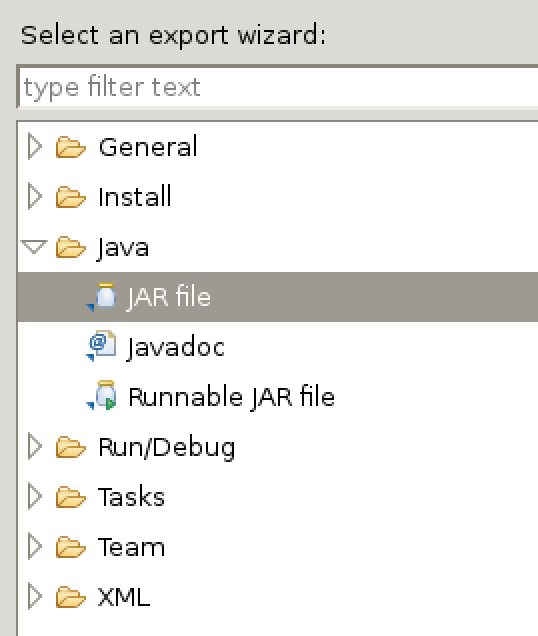
NOTE: These same steps will be used for all subsequent exercises. The instructions will not be repeated each time, so take note of the steps.

1. Verify that your Java code does not have any compiler errors or warnings.The Eclipse software in your VM is pre-‐configured to compile code automatically without performing any explicit steps. Compile errors and warnings appear as red and yellow icons to the left of the code.

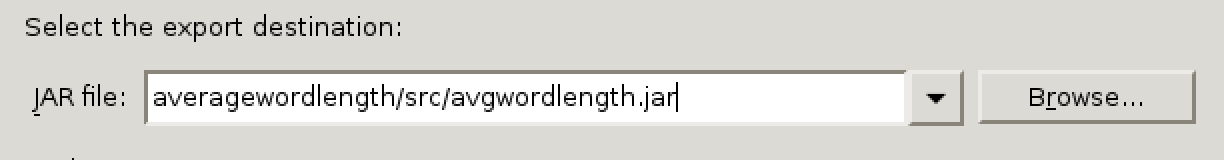


**A red X indicates a compiler error**

1. In the Package Explorer, open the Eclipse project for the current exercise (i.e.averagewordlength). Right‐click the default package under the src entry and select Export.
2. Select **Java > JAR file** from the Export dialog box, then click Next.



1. Specify a location for the **JAR file**. It works most consistently when placed in your current project folder at ~/workspace.



NOTE: Be aware that path may be different depending on your developer environment or project.

1. Test your program. In a terminal window, change to the directory where you placed your JAR file.

Run the yarn jar command as you did previously in the “Running a MapReduce Job” exercise. Make sure you use the correct package name depending on whether you are working with the provided fixme with additional hints, or just running the solution as is.

(Throughout the remainder of the exercises, the instructions will assume you are work

ing in the fixme package. Remember to replace this with the correct package name if

you are using hints or solution.)

$ yarn jar avgwordlength.jar fixme.AvgWordLength shakespeare wordlengths

1. List the results:

$ hdfs dfs -ls wordlengths

A single reducer output file should be listed.

1. Review the results:

$ hdfs dfs -cat wordlengths/\*

The file should list all the numbers and letters in the data set, and the average length of

the words starting with them, e.g.:

1.02

1.0588235294117647

1.0

1.5

1.5

1.5

1.0

1.5

1.0

3.891394576646375

5.139302507836991

6.62969423353170

…

This example uses the entire Shakespeare dataset for your input; you can also try it with just one of the files in the dataset, or with your own test data.

**Solution**

You can view the code for the solution in Eclipse in the averagewordlength/src/solution folder.

**END**