Word Co-ocurrence

-Team 1(DS)

The Team →

Prashant Pandey

Hrithik Kumar

Yashwant reddy

Jammula Supriya

Agenda →

Introduction

Problem Statement

Map Reduce

Code Implementation

Map Reduce

- Map reduce is programming model for data processing
- Framework for Parallel computing
- Allows one to to Process Huge amount of data(terabytes and petabytes) on thousands of processors

Problems → and Solutions

Problem

Word Co-occurence

what is word co-occurence?

Word co-occurrence refers to the frequency with which two or more words appear together in a given text or dataset.

Using MapReduce

Mapper phase ,Shuffle Phase, Reduce Phase

Problems → and Solutions

The text from the input text file is tokenized into words to form a key value pair with all the words present in the input text file. The key is the word from the input file and value is '1'.

After the map phase execution is completed successfully, shuffle phase is executed automatically wherein the key-value pairs generated in the map phase are taken as input and then sorted in alphabetical order.es for a particular word.

In the reduce phase, all the keys are grouped together and the values for similar keys are added up to find the occurrences for a particular word

Code → Implementation

Q1

Understanding the problem statement

Q2

Implementing the code in eclipse

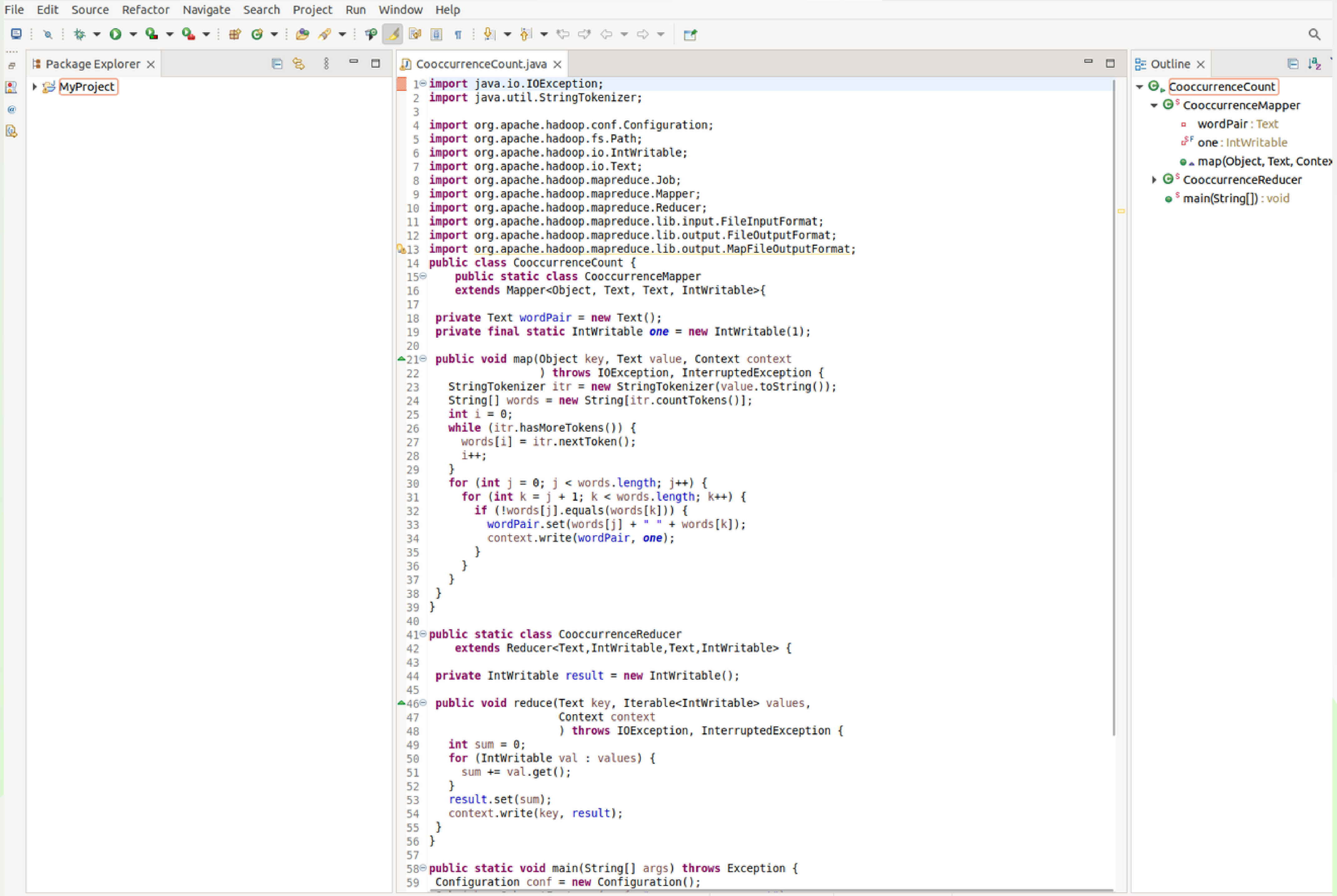
Q3

Input file

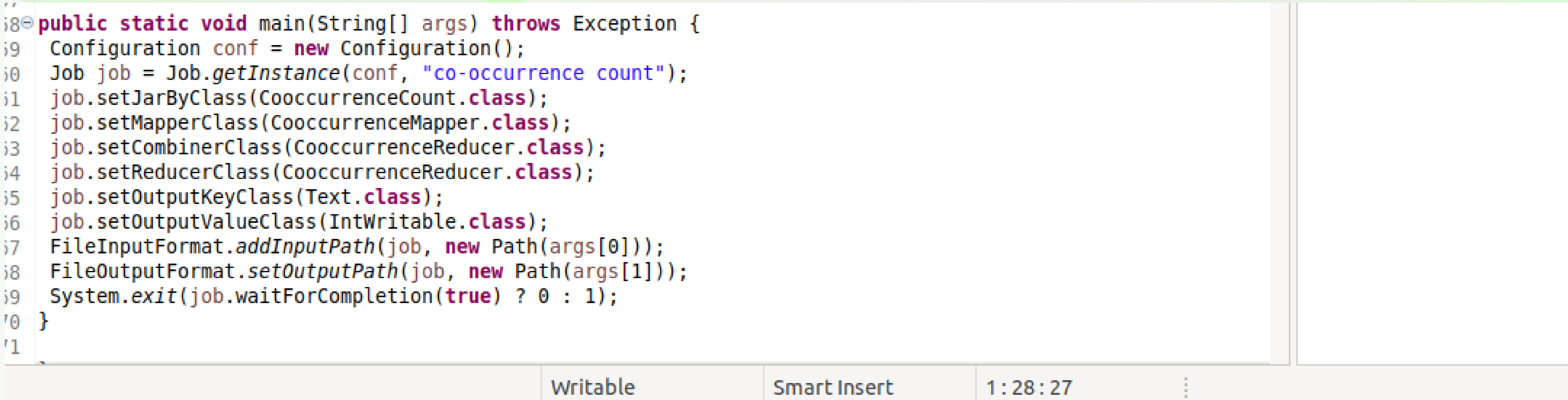
Ans

Output of the Code

Code Implementation

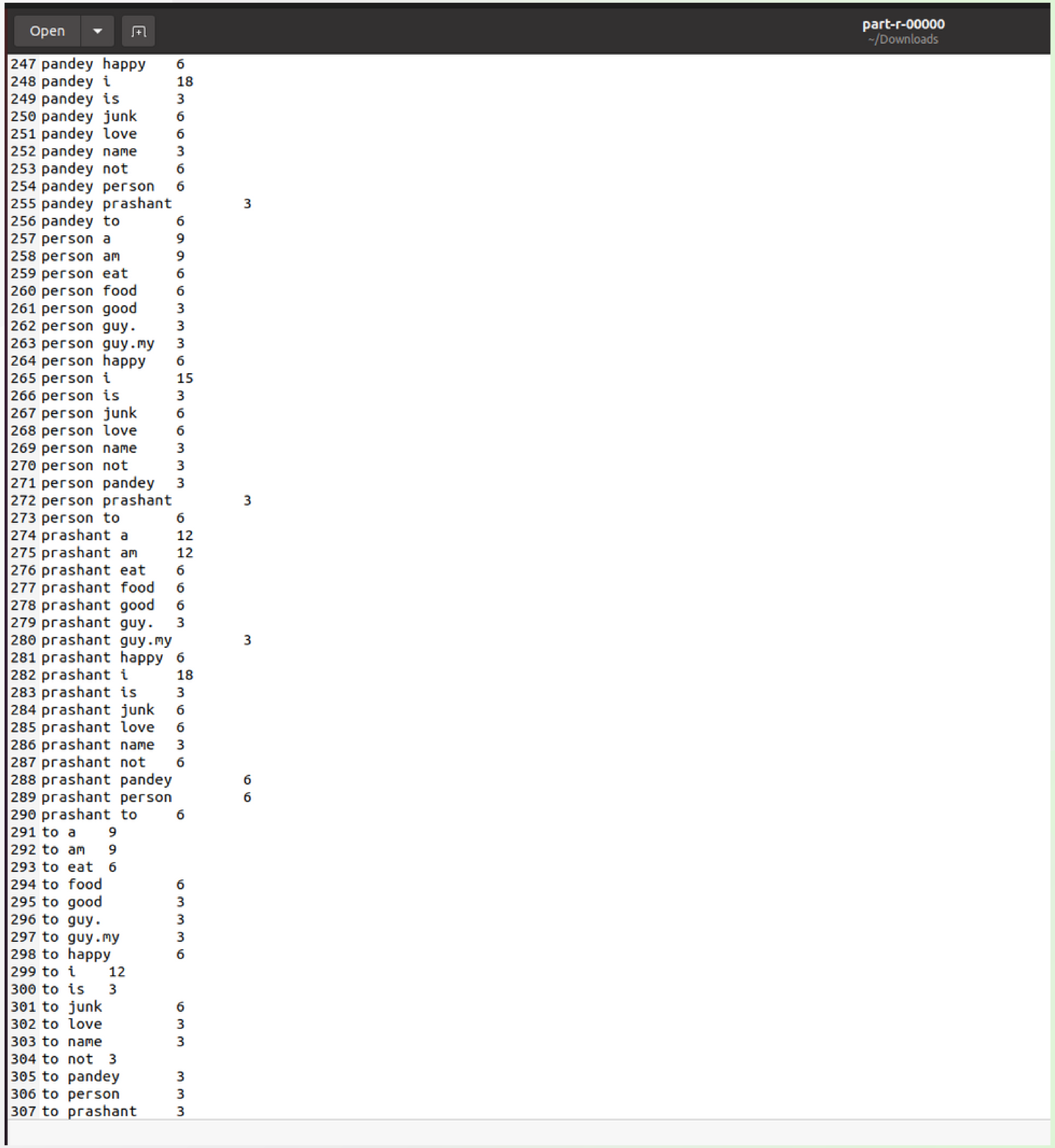


```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.MapFileOutputFormat;
public class CooccurrenceCount {
    public static class CooccurrenceMapper
    extends Mapper<Object, Text, Text, IntWritable>{

 private Text wordPair = new Text();
 private final static IntWritable one = new IntWritable(1);

 public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
   StringTokenizer itr = new StringTokenizer(value.toString());
   String[] words = new String[itr.countTokens()];
   int i = 0;
   while (itr.hasMoreTokens()) {
     words[i] = itr.nextToken();
     i++;
   }
   for (int j = 0; j < words.length; j++) {
     for (int k = j + 1; k < words.length; k++) {
       if (!words[j].equals(words[k])) {
         wordPair.set(words[j] + " " + words[k]);
         context.write(wordPair, one);
       }
     }
   }
 }
}

public static class CooccurrenceReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {

 private IntWritable result = new IntWritable();

 public void reduce(Text key, Iterable<IntWritable> values,
                   Context context
                   ) throws IOException, InterruptedException {
   int sum = 0;
   for (IntWritable val : values) {
     sum += val.get();
   }
   result.set(sum);
   context.write(key, result);
 }
}

public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
```



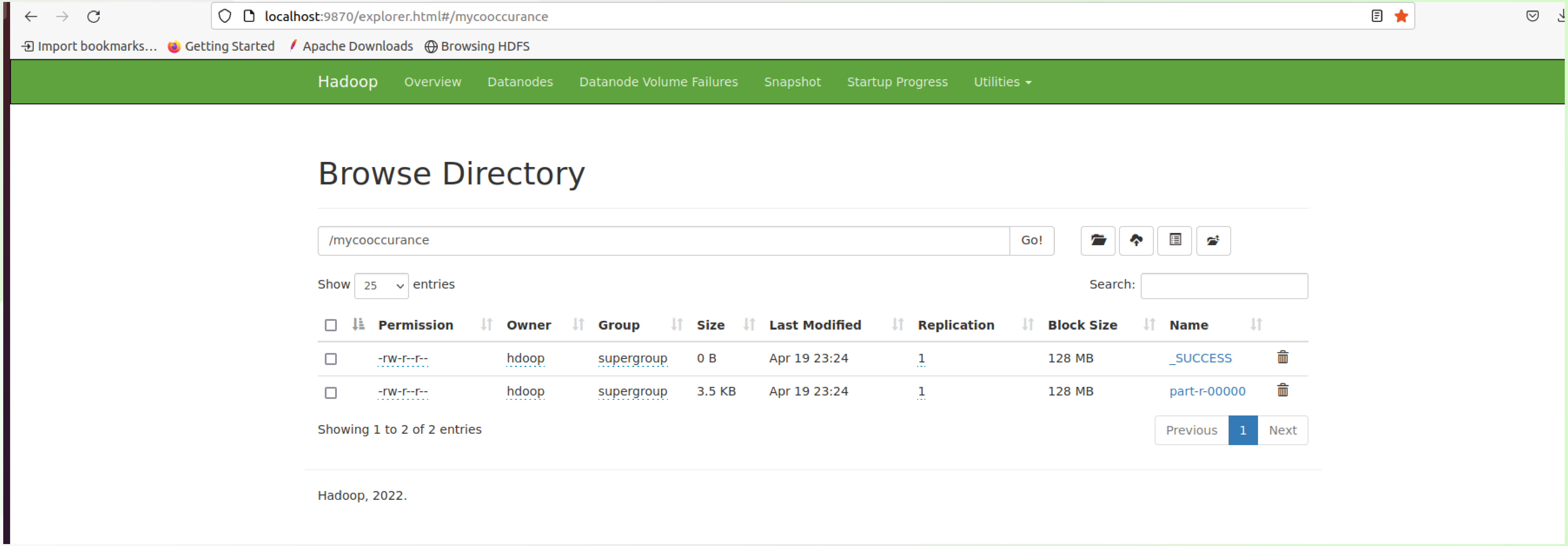
```java
public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
 Job job = Job.getInstance(conf, "co-occurrence count");
 job.setJarByClass(CooccurrenceCount.class);
 job.setMapperClass(CooccurrenceMapper.class);
 job.setCombinerClass(CooccurrenceReducer.class);
 job.setReducerClass(CooccurrenceReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 FileInputFormat.addInputPath(job, new Path(args[0]));
 FileOutputFormat.setOutputPath(job, new Path(args[1]));
 System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Output



```
pandey happy    6
pandey i        18
pandey is       3
pandey junk     6
pandey love     6
pandey name     3
pandey not      6
pandey person   6
pandey prashant         3
pandey to       6
person a        9
person am       9
person eat      6
person food     6
person good     3
person guy.     3
person guy.my   3
person happy    6
person i        15
person is       3
person junk     6
person love     6
person name     3
person not      3
person pandey   3
person prashant         3
person to       6
prashant a      12
prashant am     12
prashant eat    6
prashant food   6
prashant good   6
prashant guy.   3
prashant guy.my         3
prashant happy  6
prashant i      18
prashant is     3
prashant junk   6
prashant love   6
prashant name   3
prashant not    6
prashant pandey         6
prashant person         6
prashant to     6
to a    9
to am   9
to eat  6
to food         6
to good         3
to guy.         3
to guy.my       3
to happy        6
to i    12
to is   3
to junk         6
to love         3
to name         3
to not  3
to pandey       3
to person       3
to prashant     3
```

Output File Stored



Browse Directory

/mycooccurance

Show 25 entries

Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	hdoop	supergroup	0 B	Apr 19 23:24	1	128 MB	_SUCCESS
□	-rw-r--r--	hdoop	supergroup	3.5 KB	Apr 19 23:24	1	128 MB	part-r-00000

Showing 1 to 2 of 2 entries

Hadoop, 2022.

Thank You