// Mapper Class

**Program Two**: Frequencies of Bigrams and their Frequencies. (Only 2-words greater than 5 frequency). Driver class and Libraries are same as First Program.

**Data Preparation**: Removed Special characters.

**Output**: First and Last 50 2-words.

2. **public** **static** **class** TokenizerMapper
3. **extends** Mapper<Object, Text, Text, IntWritable>{
5. **private** **final** **static** IntWritable one = **new** IntWritable(1);
6. **private** **static** **final** Text bigram = **new** Text();
8. **public** **void** map(Object key, Text value, Context context
9. ) **throws** IOException, InterruptedException {
11. // Filtering the text for any special characters only
13. String line = value.toString().replaceAll("[^\\p{L}\\p{Z}]","");
14. String prev = **null**;
16. StringTokenizer itr = **new** StringTokenizer(line);

**while** (itr.hasMoreTokens())

           {

             String cur = itr.nextToken();

             // Emit only if we have an actual bigram.

**if** (prev != **null**)

             {

               bigram.set(prev + " " + cur);

               context.write(bigram, one);

             }

             prev = cur;

           }

          String prev1 = prev; // concatenating the first word of a line with the last word of previous line

1. }
3. }
5. // Reducer Class
7. **public** **static** **class** IntSumReducer **extends** Reducer<Text,IntWritable,Text,IntWritable> {
9. **private** IntWritable result = **new** IntWritable();
11. **public** **void** reduce(Text key, Iterable<IntWritable> values,
12. Context context
13. ) **throws** IOException, InterruptedException {
14. **int** sum = 0;
15. **for** (IntWritable val : values) {
16. sum += val.get();
17. }
18. // Filtering the words with frequency greater than 5
20. **if** (sum >5){
22. result.set(sum);
23. context.write(key, result);
24. }
26. }
28. }
30. // Driver Class
32. **public** **class** second {
34. **public** **static** **void** main(String[] args) **throws** Exception {
36. Configuration conf = **new** Configuration();
37. Job job = Job.getInstance(conf, "word count");
38. job.setJarByClass(second.**class**);
39. job.setMapperClass(TokenizerMapper.**class**);
40. job.setCombinerClass(IntSumReducer.**class**);
41. job.setReducerClass(IntSumReducer.**class**);
42. job.setOutputKeyClass(Text.**class**);
43. job.setOutputValueClass(IntWritable.**class**);
44. FileInputFormat.addInputPath(job, **new** Path(args[0]));
45. FileOutputFormat.setOutputPath(job, **new** Path(args[1]));
46. System.exit(job.waitForCompletion(**true**) ? 0 : 1);
48. }
49. }