2.

```
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;


public class wordcount {

  public static class TokenizerMapper

       extends Mapper<Object, Text, Text, IntWritable>{


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

    private Text word = new Text();


    public void map(Object key, Text value, Context context

                    ) throws IOException, InterruptedException {

      StringTokenizer itr = new StringTokenizer(value.toString());

      while (itr.hasMoreTokens()) {

        word.set(itr.nextToken());

        context.write(word, one);

      }

    }

  }

  public static class IntSumReducer

       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();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(wordcount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.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);
  }
}
```

3.

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;

public class maxtemp{
public static class TempMapper
extends Mapper<LongWritable, Text,Text,IntWritable>
private Text year=new Text();
private IntWritable temperature=new IntWritable();
@Override
protected void map(LongWritable key,Text value,Context context)
IOException,InterruptedException{
String line=value.toString();
String[] fields=line.split(" ");
year.set(fields[0]);
temperature.set9Integer.parseInt(fields[1]));
context.write(year,temperature);
}
}
public static class TempReducer extends Reducer<Text,IntWritable,Text,IntWritable>
{
private IntWritable maxTemperature=new IntWritable();
```

```
@Override

protected void reduce(Text key,Iterable<IntWritable>values,Context context)throws
IOException,Interrupted Exception{

int maxTemp=Integer.MIN_VALUE;

for(IntWritable val:values){

maxTemp=Math.max(maxTemp,val.get());

}

maxTemperature.set(maxTemp);

Context.write(key,maxTemperature);

}

}

public static void main(String[] args)throws Exception{

Configuration conf=new Configuration();

Job job=Job.getInstance(conf,"Max Temperature"):

job.setJarByClass(MaxTemperature.class);

job.setReducerClass(TempReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job,new Path(args[0]));
```

4.

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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;


public class StudentGrades {


  // Mapper class

  public static class GradeMapper extends Mapper<LongWritable, Text, Text, IntWritable> {


    private Text studentName = new Text();

    private IntWritable score = new IntWritable();


    @Override

    protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {

      String line = value.toString();

      String[] fields = line.split(" ");

      studentName.set(fields[0]); // Student name

      score.set(Integer.parseInt(fields[1])); // Score

      context.write(studentName, score); // Emit (name, score)

    }
    }


    // Reducer class
    public static class GradeReducer extends Reducer<Text, IntWritable, Text, Text> {


        private Text grade = new Text();


        @Override
        protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
            for (IntWritable value : values) {
                int score = value.get();
                // Determine grade based on scoreAJITH          A


                if (score >= 90) {
                    grade.set("A");
                } else if (score >= 80) {
                    grade.set("B");
                } else if (score >= 70) {
                    grade.set("C");
                } else if (score >= 60) {
                    grade.set("D");
                } else {
                    grade.set("F");
                }
                context.write(key, grade); // Emit (student, grade)
            }
        }
    }


    // Main driver method
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Student Grades");
        job.setJarByClass(StudentGrades.class);

        // Set Mapper and Reducer classes
        job.setMapperClass(GradeMapper.class);
        job.setReducerClass(GradeReducer.class);

        // Set output key and value types for the Mapper
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        // Set output key and value types for the Reducer
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        // Input and Output paths
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

5

```
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;
public class EvenOddCount{
  public static class EvenOddMapper extends Mapper<Object, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1); // Correct type
    private Text evenOdd = new Text();

    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      int number=Integer.parseInt(value.toString());
      if(number%2==0){
      evenOdd.set("Even");
      }else{
      evenOdd.set("odd");
      }
      context.write(evenOdd, one);
    }
  }

  public static class EvenOddReducer 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(); // Use 'val' instead of 'value'
            }
            result.set(sum); // Fixed typo 'resultr'
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Even odd count");
        job.setJarByClass(EvenOddCount.class); // Updated to match class name
        job.setMapperClass(EvenOddMapper.class);
        job.setCombinerClass(EvenOddReducer.class);
        job.setReducerClass(EvenOddReducer.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);
    }
}
```