WRITE A PARTITIONER

- Write a customised Partitioner to separate the output of weblog by years.

Files and Directories Used in this Exercise

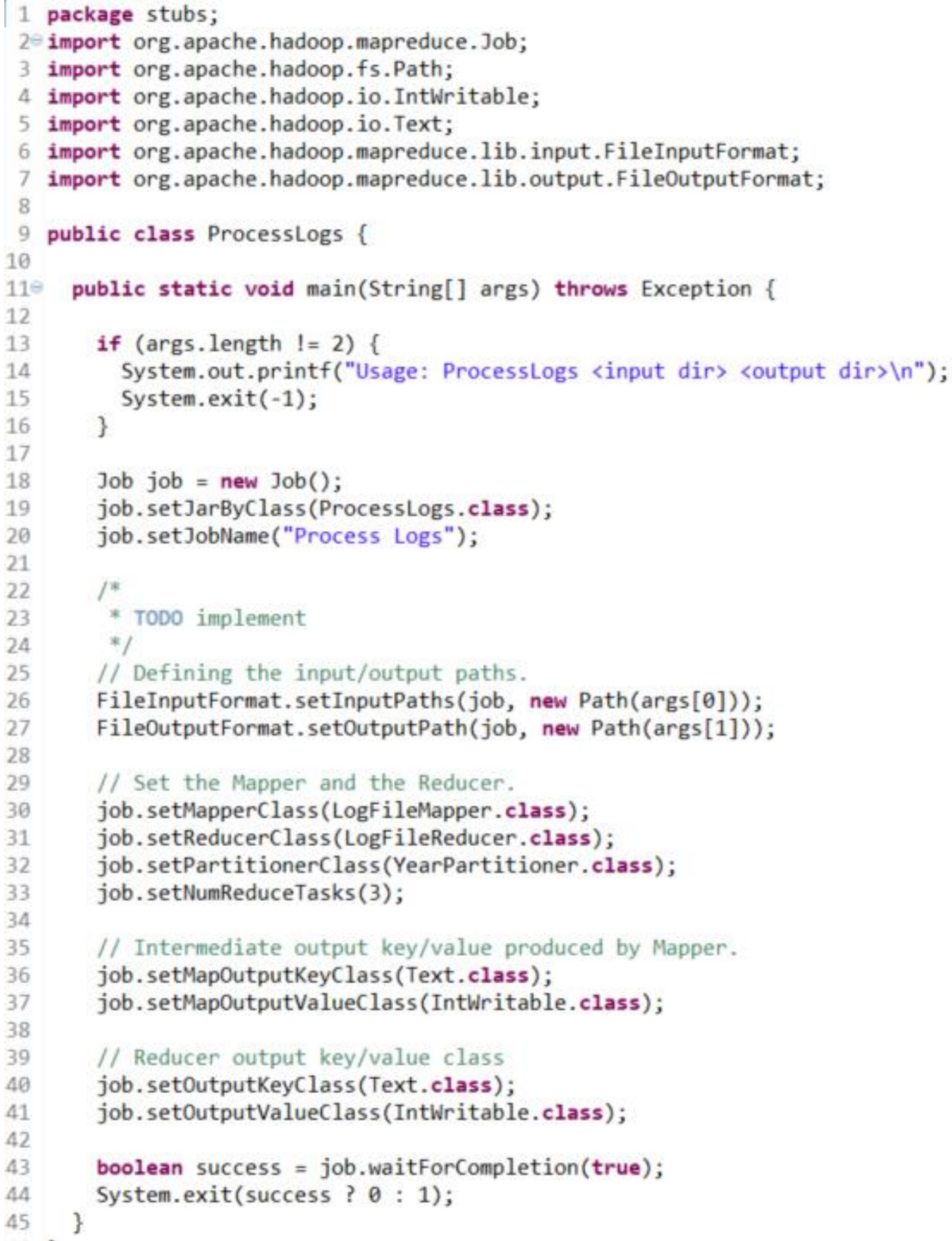
Eclipse project: partitioner

Java files: (Need to be created or copied)
YearPartitioner.java (Partitioner)
ProcessLogs.java (Driver)
LogFileMapper.java (Mapper)
LogFileReducer.java (Reducer)

Test data (HDFS):
Weblog (full web server access log)

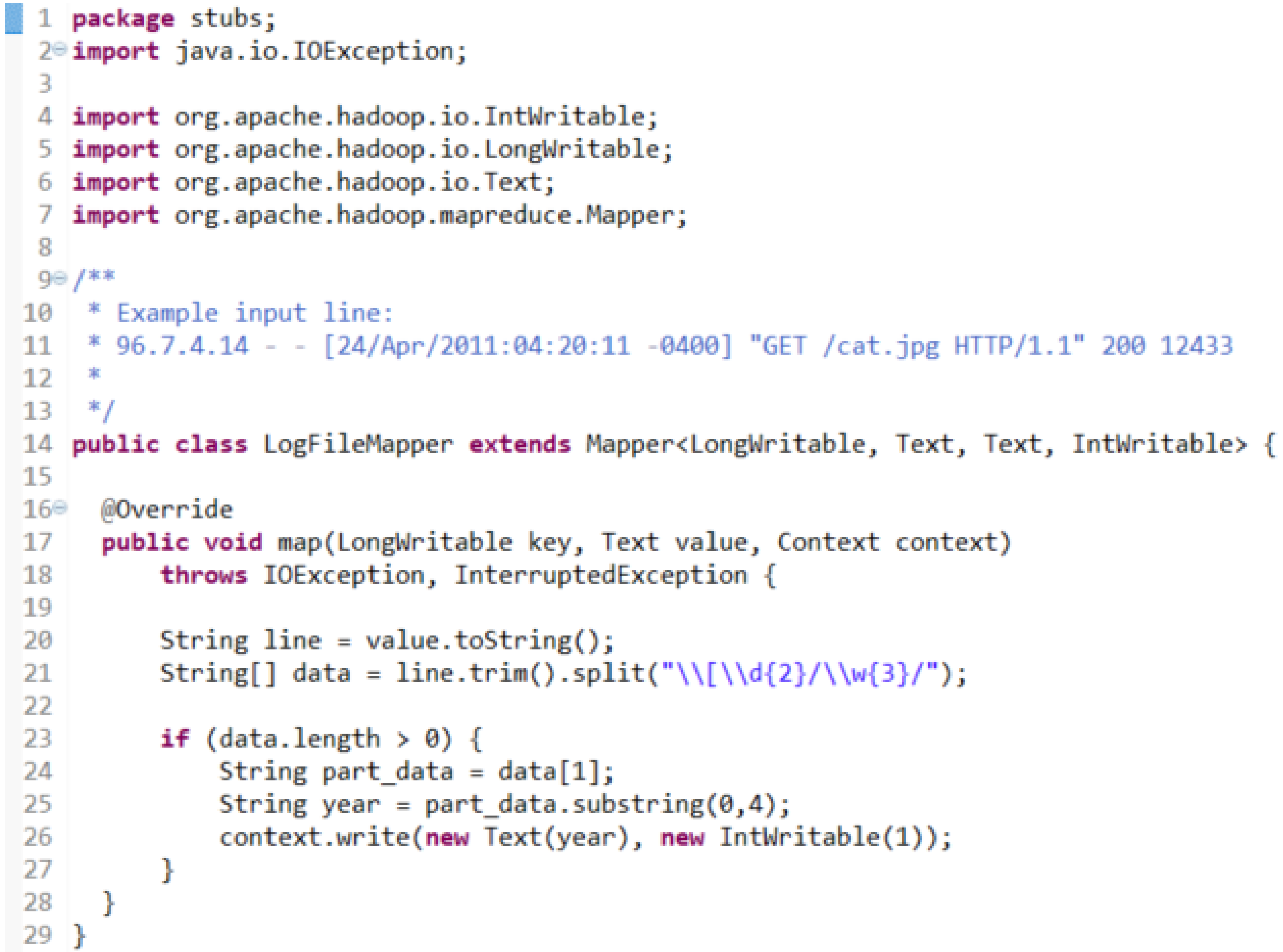
Exercise directory: ~/workspace/partitioner

Driver Code



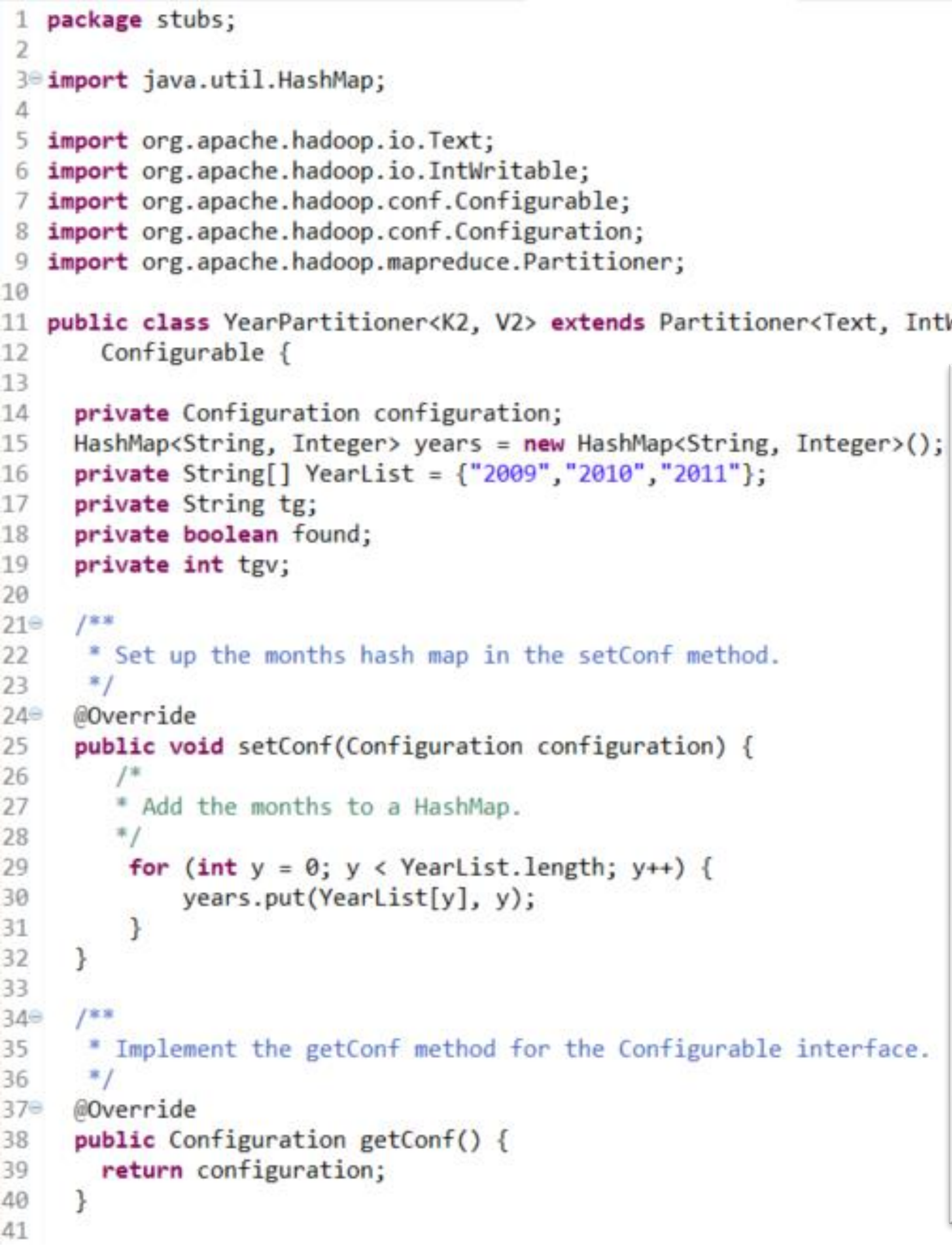
```java
package stubs;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class ProcessLogs {

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

    if (args.length != 2) {
      System.out.printf("Usage: ProcessLogs <input dir> <output dir>\n");
      System.exit(-1);
    }

    Job job = new Job();
    job.setJarByClass(ProcessLogs.class);
    job.setJobName("Process Logs");

    /*
     * TODO implement
     */
    // Defining the input/output paths.
    FileInputFormat.setInputPaths(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    // Set the Mapper and the Reducer.
    job.setMapperClass(LogFileMapper.class);
    job.setReducerClass(LogFileReducer.class);
    job.setPartitionerClass(YearPartitioner.class);
    job.setNumReduceTasks(3);

    // Intermediate output key/value produced by Mapper.
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);

    // Reducer output key/value class
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

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

MAPPER CODE

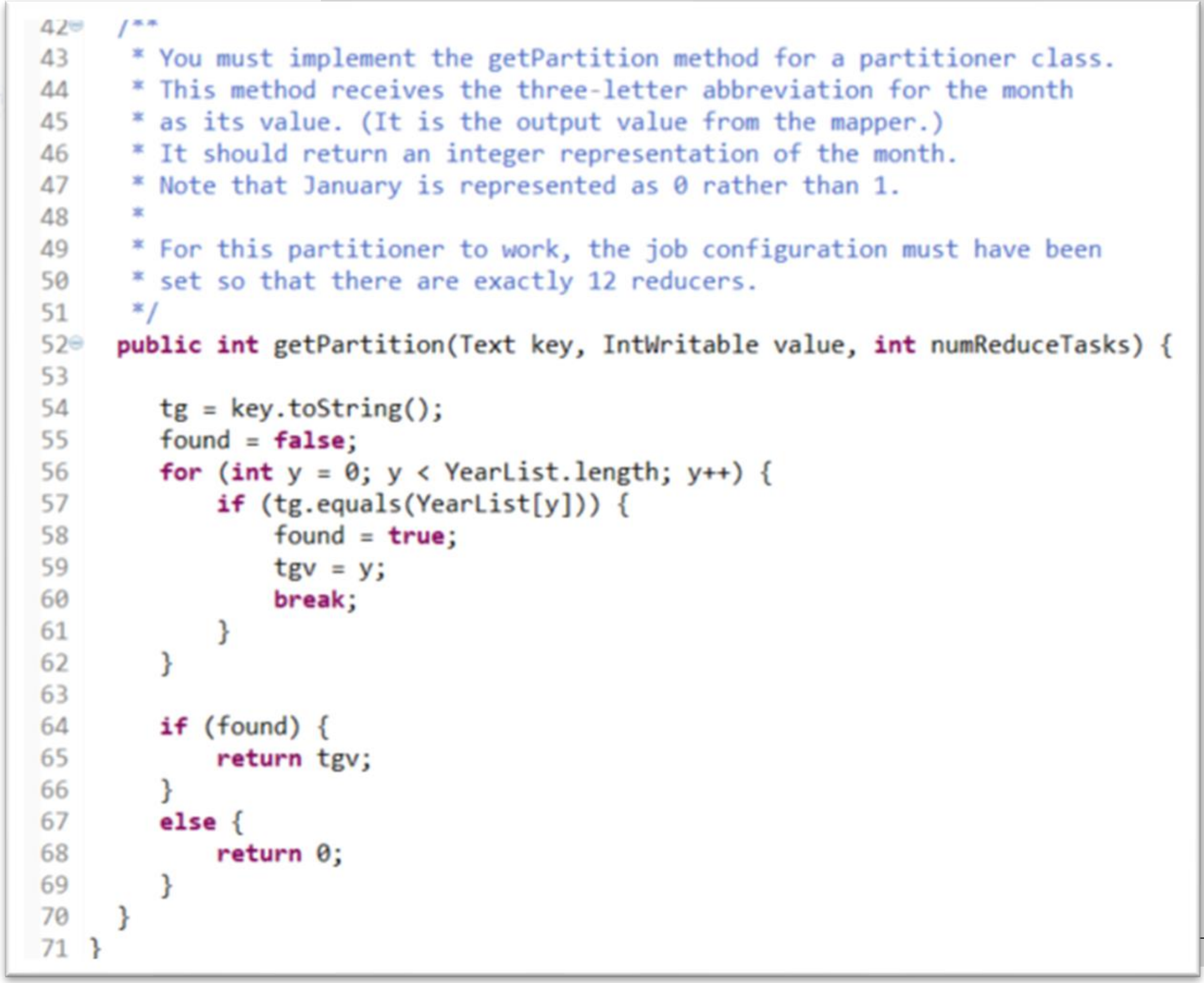


```java
package stubs;
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

/**
 * Example input line:
 * 96.7.4.14 - - [24/Apr/2011:04:20:11 -0400] "GET /cat.jpg HTTP/1.1" 200 12433
 *
 */
public class LogFileMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

  @Override
  public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {

      String line = value.toString();
      String[] data = line.trim().split("\\[\\d{2}/\\w{3}/");

      if (data.length > 0) {
          String part_data = data[1];
          String year = part_data.substring(0,4);
          context.write(new Text(year), new IntWritable(1));
      }
  }
}
```

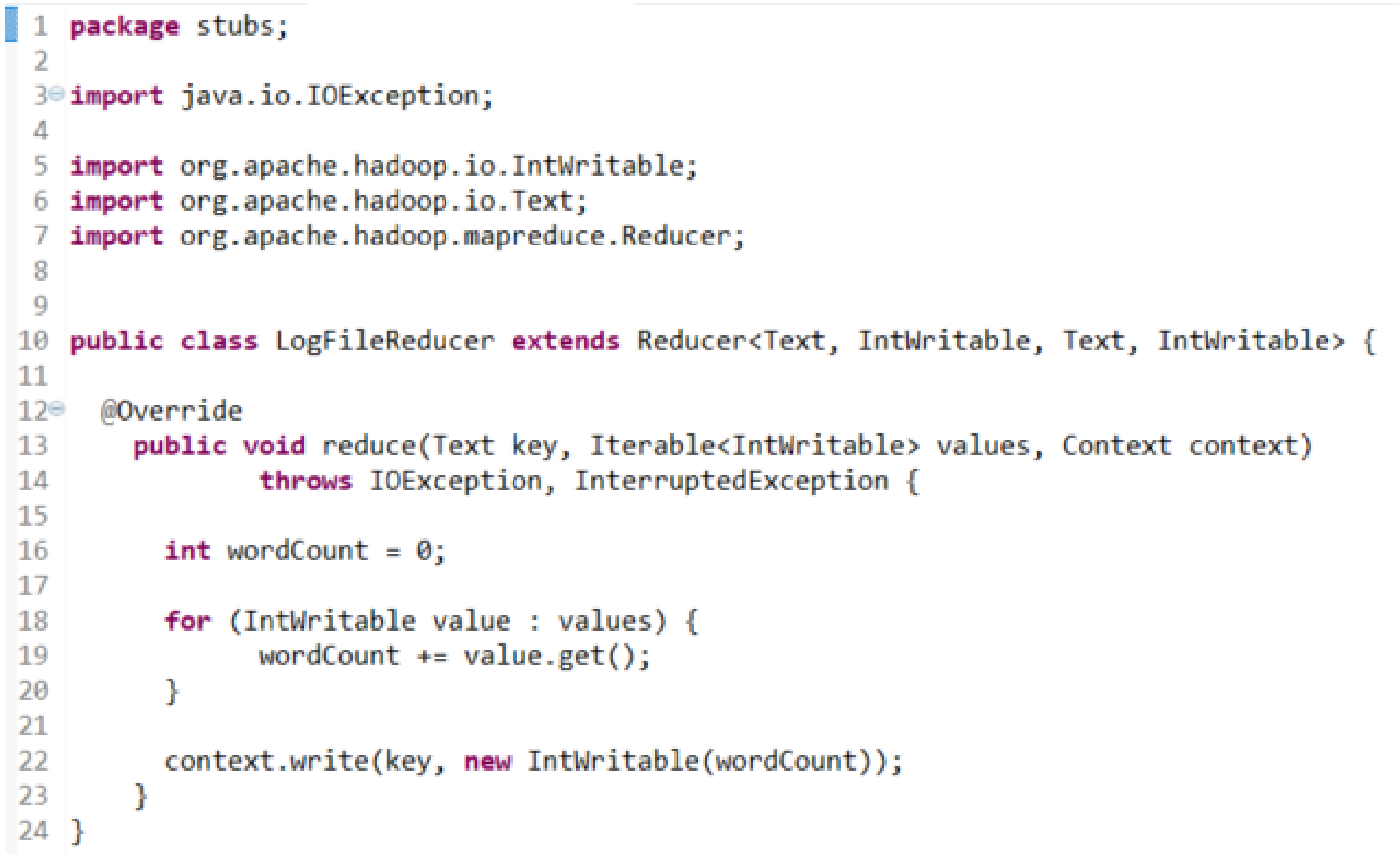
YearPartitioner Code



```java
package stubs;

import java.util.HashMap;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.conf.Configurable;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Partitioner;

public class YearPartitioner<K2, V2> extends Partitioner<Text, IntWritable> implements
    Configurable {

  private Configuration configuration;
  HashMap<String, Integer> years = new HashMap<String, Integer>();
  private String[] YearList = {"2009","2010","2011"};
  private String tg;
  private boolean found;
  private int tgv;

  /**
   * Set up the months hash map in the setConf method.
   */
  @Override
  public void setConf(Configuration configuration) {
    /*
     * Add the months to a HashMap.
     */
     for (int y = 0; y < YearList.length; y++) {
         years.put(YearList[y], y);
     }
  }

  /**
   * Implement the getConf method for the Configurable interface.
   */
  @Override
  public Configuration getConf() {
    return configuration;
  }

```

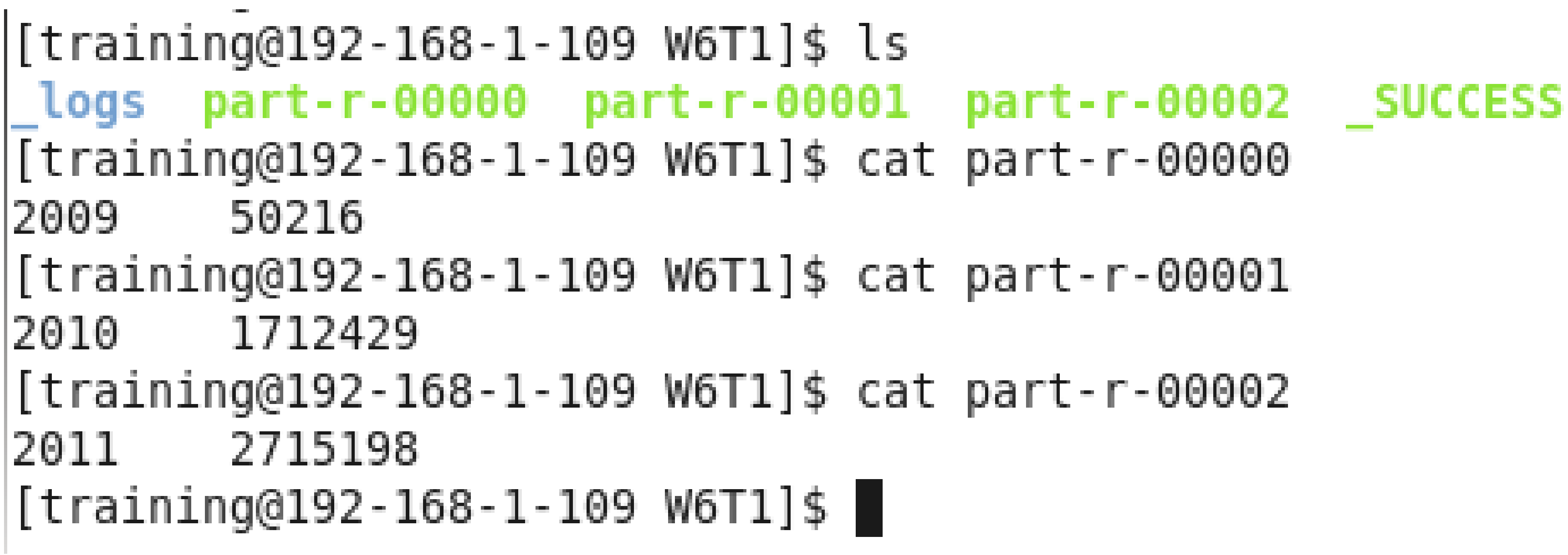


```java
  /**
   * You must implement the getPartition method for a partitioner class.
   * This method receives the three-letter abbreviation for the month
   * as its value. (It is the output value from the mapper.)
   * It should return an integer representation of the month.
   * Note that January is represented as 0 rather than 1.
   *
   * For this partitioner to work, the job configuration must have been
   * set so that there are exactly 12 reducers.
   */
  public int getPartition(Text key, IntWritable value, int numReduceTasks) {

    tg = key.toString();
    found = false;
    for (int y = 0; y < YearList.length; y++) {
        if (tg.equals(YearList[y])) {
            found = true;
            tgv = y;
            break;
        }
    }

    if (found) {
        return tgv;
    }
    else {
        return 0;
    }
  }
}
```



```java
package stubs;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;


public class LogFileReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

  @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

      int wordCount = 0;

      for (IntWritable value : values) {
            wordCount += value.get();
      }

      context.write(key, new IntWritable(wordCount));
    }
}
```

Reducer Code



```
[training@192-168-1-109 W6T1]$ ls
_logs  part-r-00000  part-r-00001  part-r-00002  _SUCCESS
[training@192-168-1-109 W6T1]$ cat part-r-00000
2009    50216
[training@192-168-1-109 W6T1]$ cat part-r-00001
2010    1712429
[training@192-168-1-109 W6T1]$ cat part-r-00002
2011    2715198
[training@192-168-1-109 W6T1]$ 
```

RESULT