作业 2

1、

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import java.io.*;
class A {
    public static boolean rm(Configuration conf, String remoteFilePath) throws IOException  {
        FileSystem fs = FileSystem.get(conf);
        Path remotePath = new Path(remoteFilePath);
        boolean result = fs.delete(remotePath, false);
        fs.close();
        return result;
     }
    public static boolean mv(Configuration conf, String remoteFilePath, String remoteToFilePath)
          throws IOException {
        FileSystem fs = FileSystem.get(conf);
        Path srcPath = new Path(remoteFilePath);
        Path dstPath = new Path(remoteToFilePath);
        boolean result = fs.rename(srcPath, dstPath);
        fs.close();
        return result;
     }
}
public class B  {
     public static void main(String[] args)  {
           Configuration conf = new Configuration();  conf.set("fs.default.name","hdfs://localhost:9000");
        String remoteFilePath = "/user/hadoop/text.txt";
        try {
              if ( A.rm(conf, remoteFilePath) )
                 {  System.out.println("文件删除: " + remoteFilePath);  }
              else {  System.out.println("操作失败（文件不存在或删除失败）");  }
        }
        catch (Exception e)
              { e.printStackTrace(); }
     String remoteFilePath1 = "/user/hadoop/text1.txt";
        String remoteToFilePath1 = "/user/hadoop/new.txt";
        try {
            if ( A.mv(conf, remoteFilePath1, remoteToFilePath1) ) {
              System.out.println("将文件" +remoteFilePath1+"移动到" + remoteToFilePath1);
            }
            else
            { System.out.println("操作失败(源文件不存在或移动失败)"); }
        }
```

```
            catch (Exception e)
                  { e.printStackTrace(); }
      }
}
```

2、

```
      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.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 MaxValue  {
         private static int mapnum=0;
         private static int reducenum=0;
         public static void main(String[] args) throws Exception {
            Configuration conf = new Configuration();
            conf.set("fs.defaultFS", "hdfs://localhost:9000");
            String[] otherArgs = new String[]{"input","output"};
            if(otherArgs.length < 2) {
                System.err.println("Usage: wordcount <in> [<in>...] <out>");
                System.exit(2);
            }
            Job job = Job.getInstance(conf, "MaxValue");
            job.setJarByClass(MaxValue.class);
            job.setMapperClass(Map.class);
            job.setReducerClass(Reduce.class);
            job.setMapOutputKeyClass(IntWritable.class);
            job.setMapOutputValueClass(IntWritable.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
            FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
            System.exit(job.waitForCompletion(true)? 0:1);
        }
        public static class Map extends Mapper<Object, Text, IntWritable, IntWritable> {
            private IntWritable data = new IntWritable();
                    public void map(Object key, Text value, Context context) throws IOException,
                                    InterruptedException
                 {
```

```
            String t=value.toString();
            data.set(Integer.parseInt(t));
            context.write(data, new IntWritable(1));
            mapnum++;
        }
    }
    public static class Reduce extends Reducer<IntWritable, IntWritable,Text,IntWritable>{
                public void reduce(IntWritable key, Iterable<IntWritable> values, Context context) throws
                                IOException, InterruptedException
            {
            for(IntWritable val:values)
            {
             reducenum++;
            }
            if(reducenum==mapnum) context.write(new Text("最大值："),key);
        }
    }
}
```