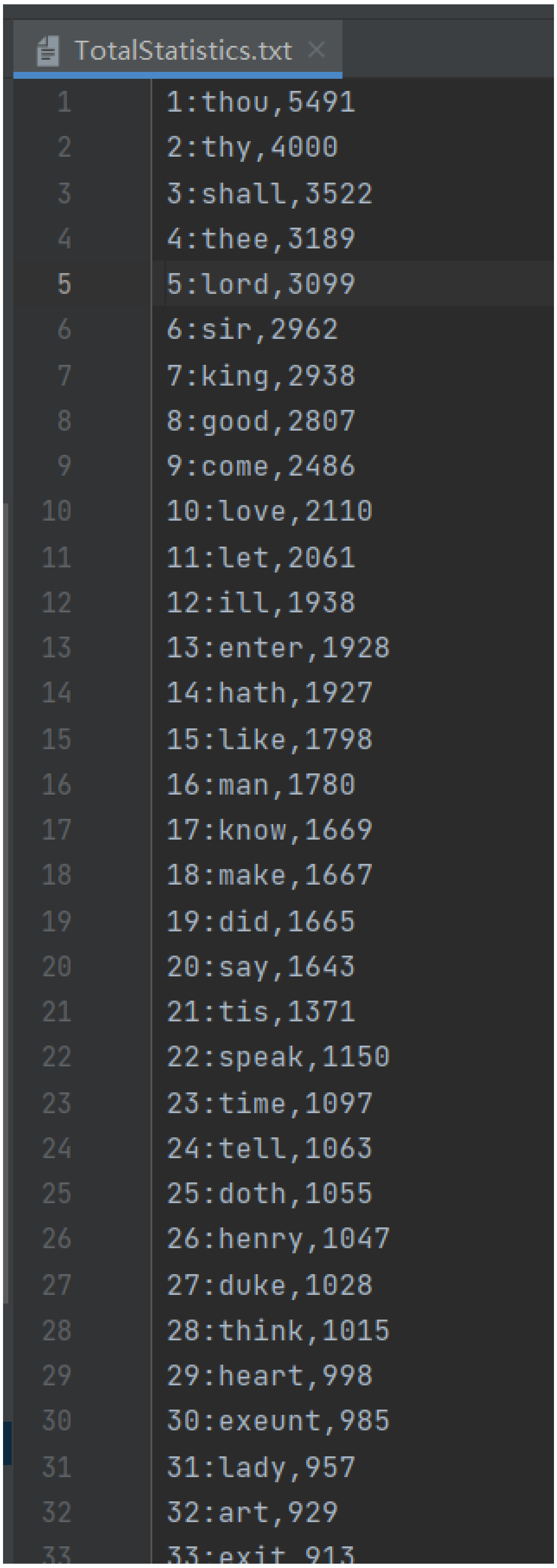
《金融大数据处理技术》作业5汇报

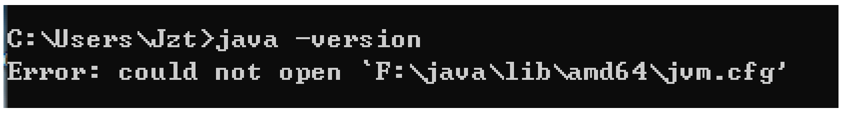
191870068 嵇泽同

运行结果：



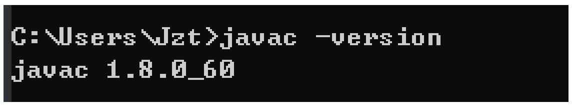
```
1:thou,5491
2:thy,4000
3:shall,3522
4:thee,3189
5:lord,3099
6:sir,2962
7:king,2938
8:good,2807
9:come,2486
10:love,2110
11:let,2061
12:ill,1938
13:enter,1928
14:hath,1927
15:like,1798
16:man,1780
17:know,1669
18:make,1667
19:did,1665
20:say,1643
21:tis,1371
22:speak,1150
23:time,1097
24:tell,1063
25:doth,1055
26:henry,1047
27:duke,1028
28:think,1015
29:heart,998
30:exeunt,985
31:lady,957
32:art,929
33:exit,913
```

JAVA安装多版本后，即使修改了环境变量，cmd中依然找不到正确的路径：



```
C:\Users\Jzt>java -version
Error: could not open `F:\java\lib\amd64\jvm.cfg'
```

但是javac -version可以查看到jdk1.8



```
C:\Users\Jzt>javac -version
javac 1.8.0_60
```

应该还是环境变量配置的问题，上网查找后解决：

https://blog.csdn.net/KingJin_CSDN_/article/details/53667319?spm=1001.2101.3001.6650.5&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EOPENSEARCH%7Edefault-5.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EOPENSEARCH%7Edefault-5.no_search_link

1、出现原因

我之前安装的jdk是1.6；后来又装成1.8，把变量名JAVA_HOME改成了1.8的文件路径，最终导致java -version和javac -version显示的版本不一致。

2、错误结果

java -version 显示的是最新安装版本的java
javac -version 显示的是你配置环境变量版本的java

3、修改方法

把 %JAVA_HOME% 放在Path的头部；

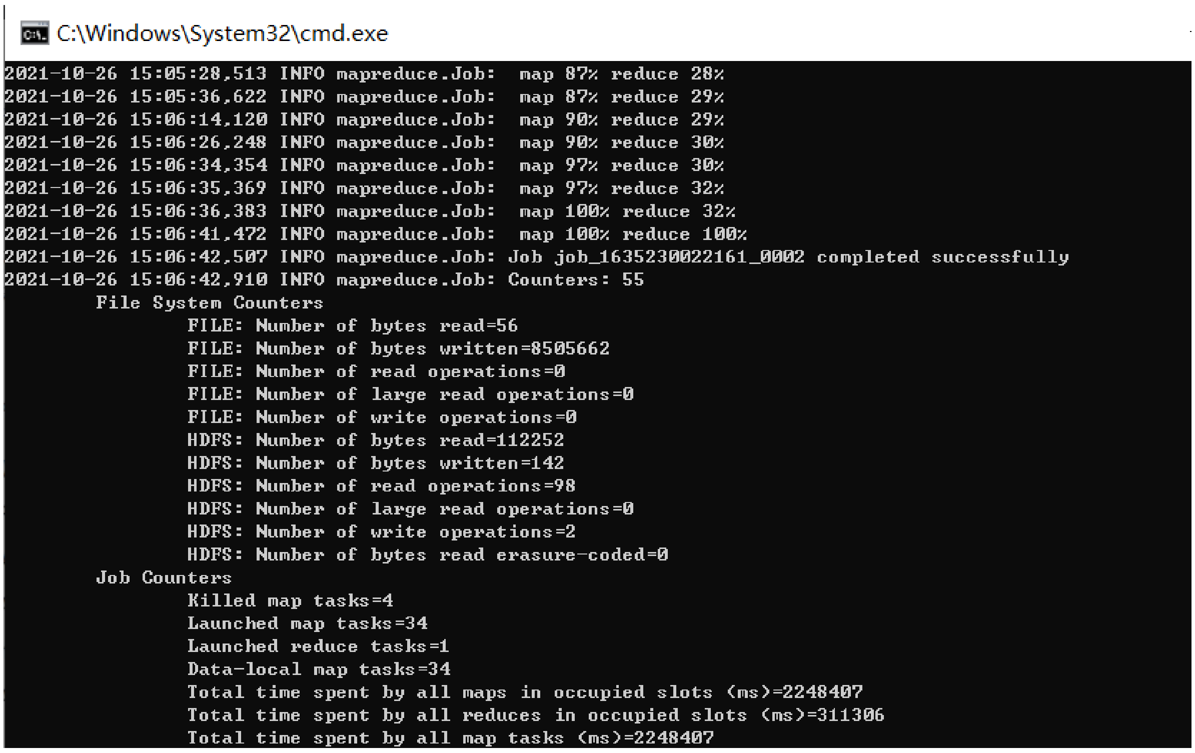
1. 变量名：JAVA_HOME
2. 变量值：C:\Program Files\Java\jdk1.8.0
3. 变量名：CLASSPATH
4. 变量值：%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;
5. 变量名：Path
6. 变量值：%JAVA_HOME%\bin;

这些变量都是配置在系统变量里面的（见下图）

win10下安装hadoop教程：

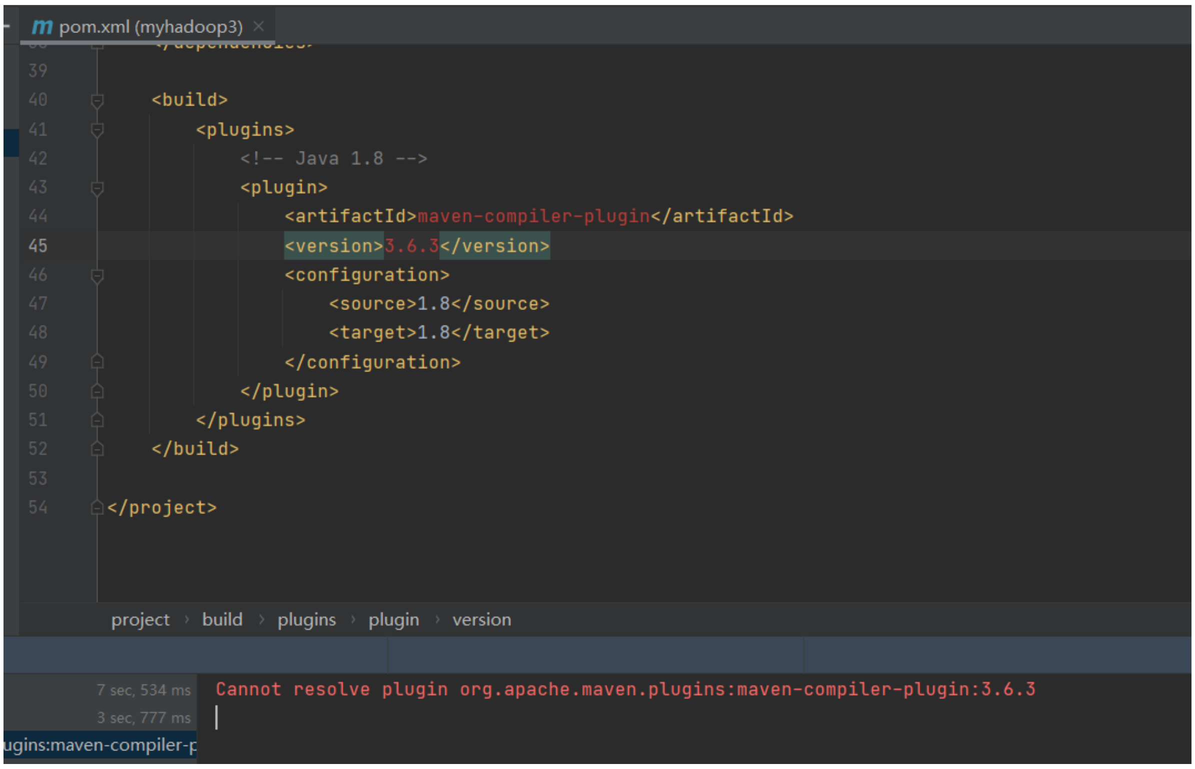
https://zhuanlan.zhihu.com/p/375066643

运行grep样例，成功：



C:\Windows\System32\cmd.exe

```
2021-10-26 15:05:28,513 INFO mapreduce.Job:  map 87% reduce 28%
2021-10-26 15:05:36,622 INFO mapreduce.Job:  map 87% reduce 29%
2021-10-26 15:06:14,120 INFO mapreduce.Job:  map 90% reduce 29%
2021-10-26 15:06:26,248 INFO mapreduce.Job:  map 90% reduce 30%
2021-10-26 15:06:34,354 INFO mapreduce.Job:  map 97% reduce 30%
2021-10-26 15:06:35,369 INFO mapreduce.Job:  map 97% reduce 32%
2021-10-26 15:06:36,383 INFO mapreduce.Job:  map 100% reduce 32%
2021-10-26 15:06:41,472 INFO mapreduce.Job:  map 100% reduce 100%
2021-10-26 15:06:42,507 INFO mapreduce.Job: Job job_1635230022161_0002 completed successfully
2021-10-26 15:06:42,910 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=56
                FILE: Number of bytes written=8505662
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=112252
                HDFS: Number of bytes written=142
                HDFS: Number of read operations=98
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=4
                Launched map tasks=34
                Launched reduce tasks=1
                Data-local map tasks=34
                Total time spent by all maps in occupied slots (ms)=2248407
                Total time spent by all reduces in occupied slots (ms)=311306
                Total time spent by all map tasks (ms)=2248407
```

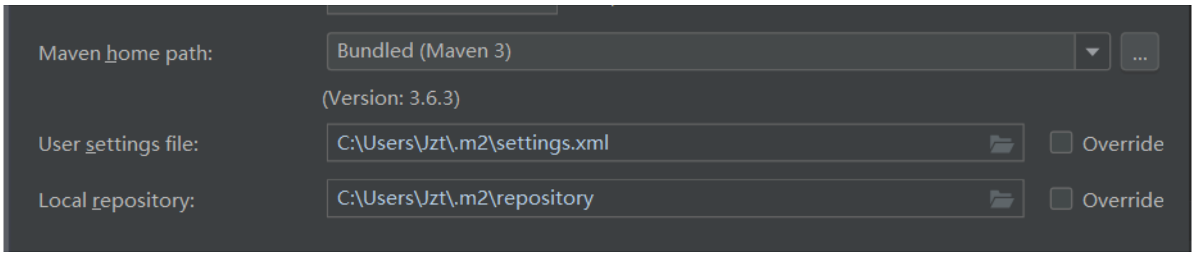


pom.xml (myhadoop3)

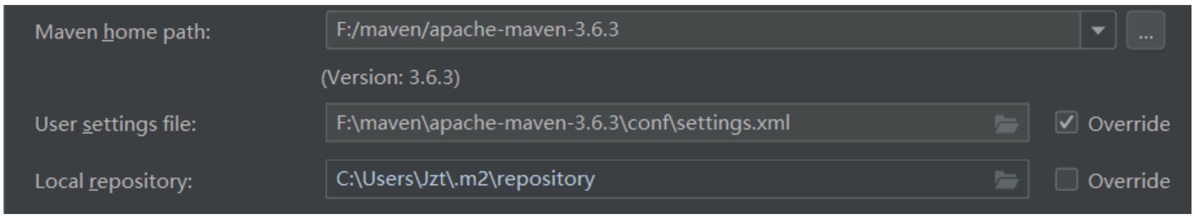
```xml

    <build>
        <plugins>
            <!-- Java 1.8 -->
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.6.3</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

project › build › plugins › plugin › version

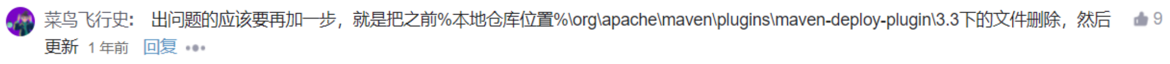
```
7 sec, 534 ms    Cannot resolve plugin org.apache.maven.plugins:maven-compiler-plugin:3.6.3
3 sec, 777 ms
ugins:maven-compiler-p
```

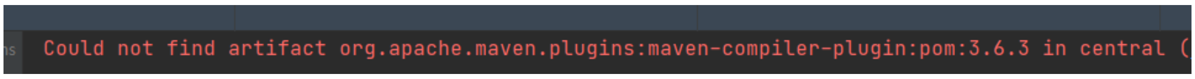


Maven home path:	Bundled (Maven 3) (Version: 3.6.3)	
User settings file:	C:\Users\Jzt\.m2\settings.xml	☐ Override
Local repository:	C:\Users\Jzt\.m2\repository	☐ Override



Maven home path:	F:/maven/apache-maven-3.6.3 (Version: 3.6.3)	
User settings file:	F:\maven\apache-maven-3.6.3\conf\settings.xml	✓ Override
Local repository:	C:\Users\Jzt\.m2\repository	☐ Override

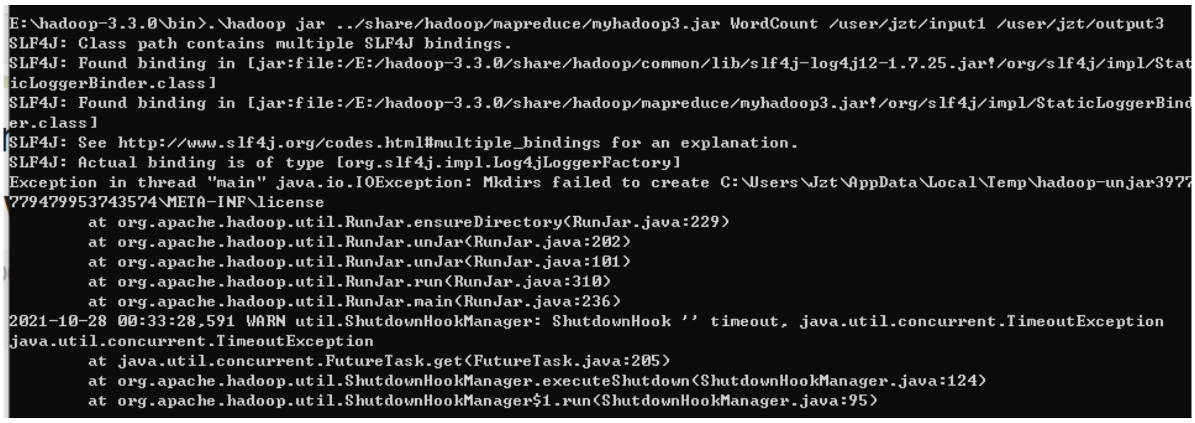

菜鸟飞行史：出问题的应该要再加一步，就是把之前%本地仓库位置%\org\apache\maven\plugins\maven-deploy-plugin\3.3下的文件删除，然后更新 1 年前 回复 ••• 9



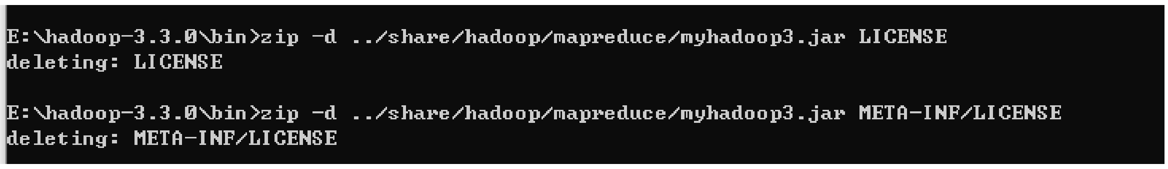
```
Could not find artifact org.apache.maven.plugins:maven-compiler-plugin:pom:3.6.3 in central (
```

怎么都搞不好，上网查了这个插件的作用后觉得不需要也行，就直接删了。

运行出错：



```
E:\hadoop-3.3.0\bin>.\hadoop jar ../share/hadoop/mapreduce/myhadoop3.jar WordCount /user/jzt/input1 /user/jzt/output3
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/E:/hadoop-3.3.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/E:/hadoop-3.3.0/share/hadoop/mapreduce/myhadoop3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Exception in thread "main" java.io.IOException: Mkdirs failed to create C:\Users\Jzt\AppData\Local\Temp\hadoop-unjar3977
779479953743574\META-INF\license
        at org.apache.hadoop.util.RunJar.ensureDirectory(RunJar.java:229)
        at org.apache.hadoop.util.RunJar.unJar(RunJar.java:202)
        at org.apache.hadoop.util.RunJar.unJar(RunJar.java:101)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:310)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
2021-10-28 00:33:28,591 WARN util.ShutdownHookManager: ShutdownHook '' timeout, java.util.concurrent.TimeoutException
java.util.concurrent.TimeoutException
        at java.util.concurrent.FutureTask.get(FutureTask.java:205)
        at org.apache.hadoop.util.ShutdownHookManager.executeShutdown(ShutdownHookManager.java:124)
        at org.apache.hadoop.util.ShutdownHookManager$1.run(ShutdownHookManager.java:95)
```

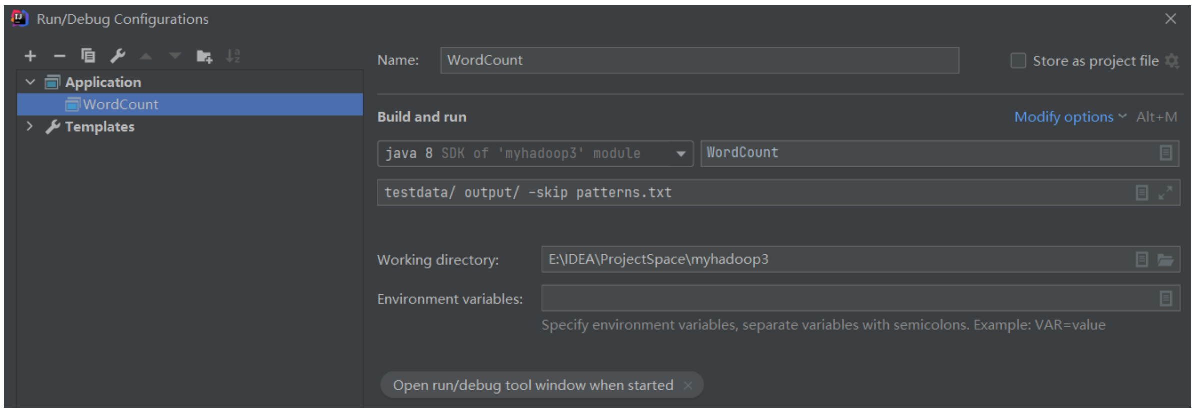


```
E:\hadoop-3.3.0\bin>zip -d ../share/hadoop/mapreduce/myhadoop3.jar LICENSE
deleting: LICENSE

E:\hadoop-3.3.0\bin>zip -d ../share/hadoop/mapreduce/myhadoop3.jar META-INF/LICENSE
deleting: META-INF/LICENSE
```

参考的IDEA编写hadoop程序教程：

https://www.cnblogs.com/ziyaziya/p/12456586.html

后来发现不用打成jar包在命令行里运行，直接在IDEA里就可以运行，速度很快，也方便DEBUG，配置好后右键运行程序即可：



Name: WordCount

java 8 SDK of 'myhadoop3' module | WordCount

testdata/ output/ -skip patterns.txt

Working directory: E:\IDEA\ProjectSpace\myhadoop3

Environment variables:

Specify environment variables, separate variables with semicolons. Example: VAR=value

共可分为**6**个详细的阶段：

1).Collect阶段：将MapTask的结果输出到默认大小为100M的MapOutputBuffer内部环形内存缓冲区，保存的是key/value，Partition分区

2).Spill阶段：当内存中的数据量达到一定的阀值的时候，就会将数据写入本地磁盘，在将数据写入磁盘之前需要对数据进行一次排序的操作，先是对partition分区号进行排序，再对key排序，如果配置了combiner，还会将有相同分区号和key的数据进行排序，如果有压缩设置，则还会对数据进行压缩操作。

3).Combiner阶段：等MapTask任务的数据处理完成之后，会对所有map产生的数据结果进行一次合并操作，以确保一个MapTask最终只产生一个中间数据文件。

4).Copy阶段：当整个MapReduce作业的MapTask所完成的任务数据占到MapTask总数的5%时，JobTracker就会调用ReduceTask启动，此时ReduceTask就会默认的启动5个线程到已经完成MapTask的节点上复制一份属于自己的数据，这些数据默认会保存在内存的缓冲区中，当内存的缓冲区达到一定的阀值的时候，就会将数据写到磁盘之上。

5).Merge阶段：在ReduceTask远程复制数据的同时，会在后台开启两个线程对内存中和本地中的数据文件进行合并操作。

6).Sort阶段：在对数据进行合并的同时，会进行排序操作，由于MapTask阶段已经对数据进行了局部的排序，ReduceTask只需做一次归并排序就可以保证Copy的数据的整体有效性。

用一个MAP-REDUCE过程感觉很难实现，因为在PARTITIONER分配KEY时是不知道每个词的词频的，后续在SHUFFLE过程中即使通过combiner统计出词频，也无法再次重新PARTITIONER分配，因此每个REDUCE节点得到的数据在词频上可能有很大差异。因此考虑采用链式MAPREDUCE。

修改Reduce的输出格式后，map中context.write()报错：Type mismatch in key from map: expected org.apache.hadoop.io.IntWritable, received org.apache.hadoop.io.IntWritable。

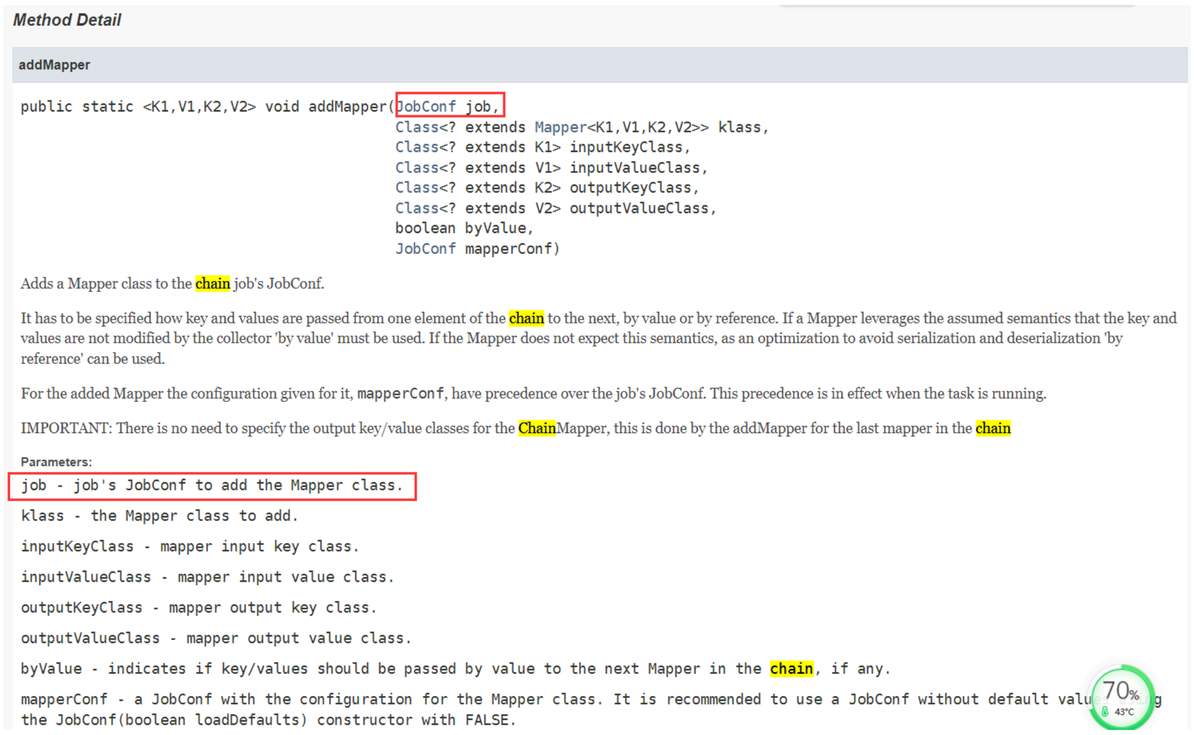
但是我的map的输出类型和reduce的输入类型一致，为什么会报这个错？

后来反复上网查找、尝试，发现需要添加如下两行，显示指定map的输出格式：

```
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
```

不添加这两行的话，即使在Mapper类和Reduce类的参数中指定一致的输出格式也会报错，网上有解释是当map方法中context.write()和reduce方法中context.write()输入参数类型不相同时，需要在job中设置每个方法的参数类型。

本来打算用Chain，但是在API参数上遇到了问题。官网：



Method Detail

addMapper

```
public static <K1,V1,K2,V2> void addMapper(JobConf job,
                                           Class<? extends Mapper<K1,V1,K2,V2>> klass,
                                           Class<? extends K1> inputKeyClass,
                                           Class<? extends V1> inputValueClass,
                                           Class<? extends K2> outputKeyClass,
                                           Class<? extends V2> outputValueClass,
                                           boolean byValue,
                                           JobConf mapperConf)
```

Adds a Mapper class to the chain job's JobConf.

It has to be specified how key and values are passed from one element of the chain to the next, by value or by reference. If a Mapper leverages the assumed semantics that the key and values are not modified by the collector 'by value' must be used. If the Mapper does not expect this semantics, as an optimization to avoid serialization and deserialization 'by reference' can be used.

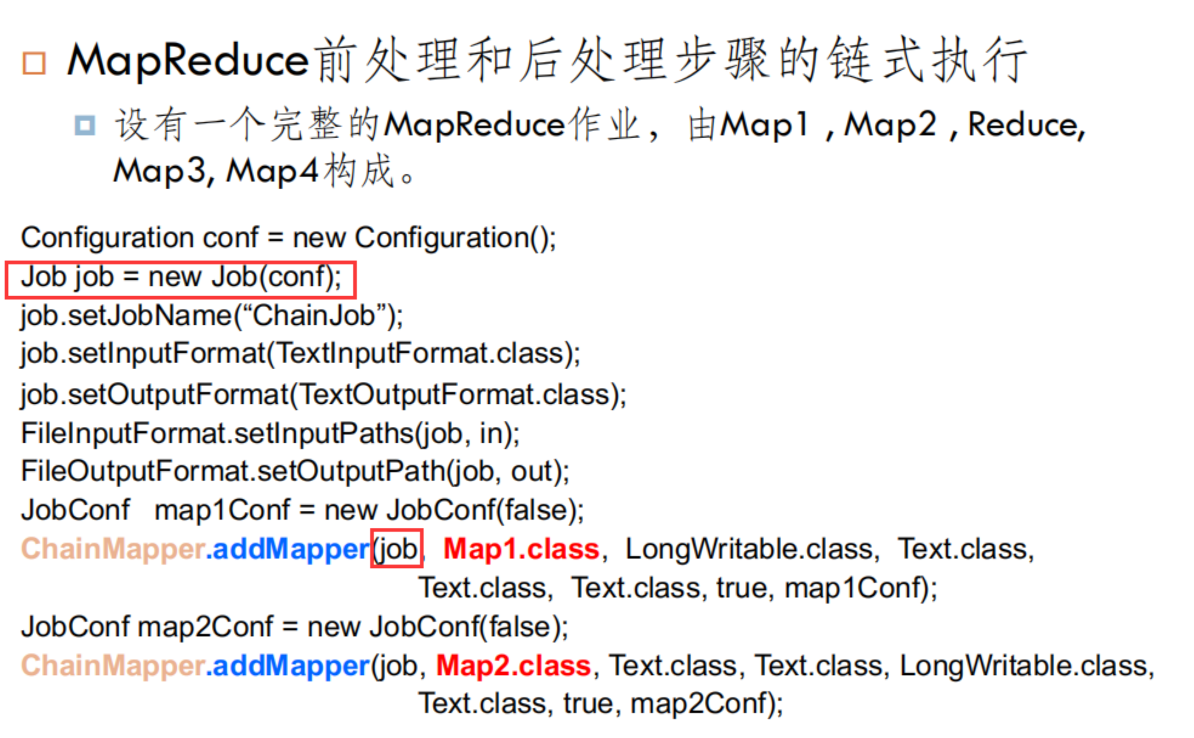
For the added Mapper the configuration given for it, `mapperConf`, have precedence over the job's JobConf. This precedence is in effect when the task is running.

IMPORTANT: There is no need to specify the output key/value classes for the ChainMapper, this is done by the addMapper for the last mapper in the chain

Parameters:

```
job - job's JobConf to add the Mapper class.
klass - the Mapper class to add.
inputKeyClass - mapper input key class.
inputValueClass - mapper input value class.
outputKeyClass - mapper output key class.
outputValueClass - mapper output value class.
byValue - indicates if key/values should be passed by value to the next Mapper in the chain, if any.
mapperConf - a JobConf with the configuration for the Mapper class. It is recommended to use a JobConf without default valu[illegible]g
the JobConf(boolean loadDefaults) constructor with FALSE.
```

PPT：



MapReduce前处理理和后处理理步骤的链式执行

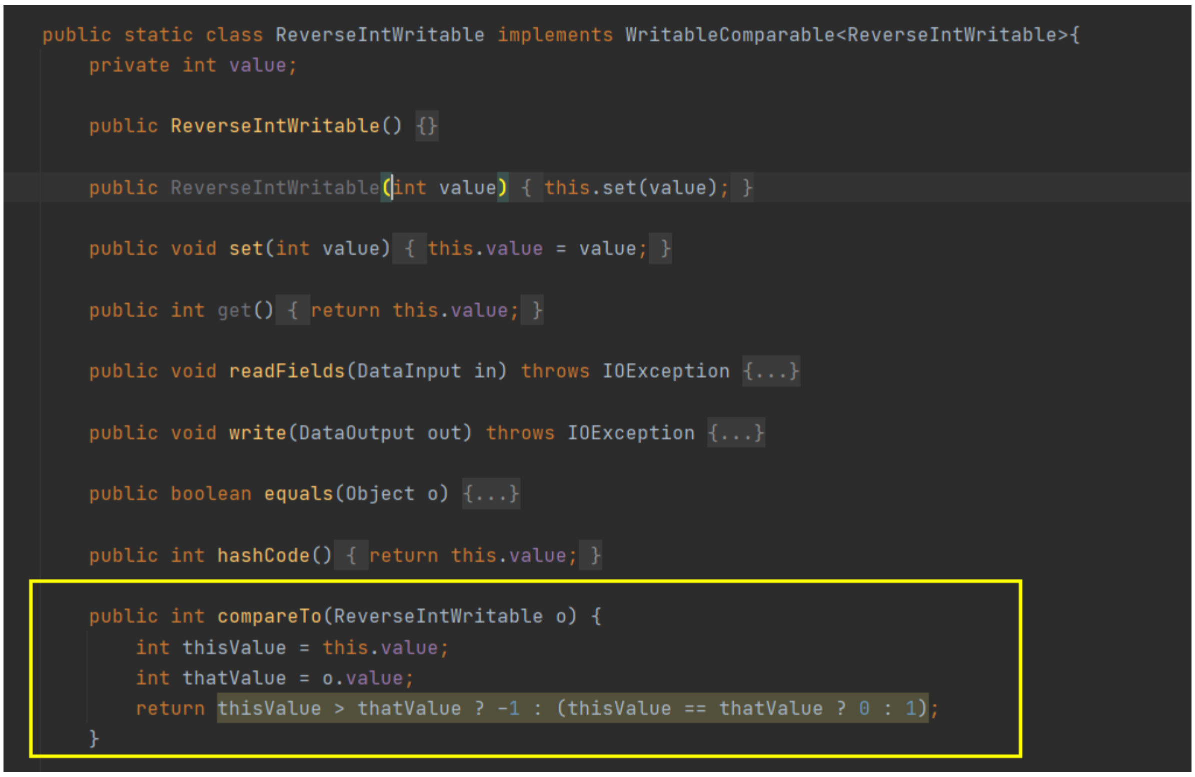
- 设有一个完整的MapReduce作业，由Map1，Map2，Reduce, Map3, Map4构成。

```
Configuration conf = new Configuration();
Job job = new Job(conf);
job.setJobName("ChainJob");
job.setInputFormat(TextInputFormat.class);
job.setOutputFormat(TextOutputFormat.class);
FileInputFormat.setInputPaths(job, in);
FileOutputFormat.setOutputPath(job, out);
JobConf  map1Conf = new JobConf(false);
ChainMapper.addMapper(job, Map1.class, LongWritable.class, Text.class,
                      Text.class, Text.class, true, map1Conf);
JobConf map2Conf = new JobConf(false);
ChainMapper.addMapper(job, Map2.class, Text.class, Text.class, LongWritable.class,
                      Text.class, true, map2Conf);
```

而且不断尝试后发现不知道如何从Job类的实例job获得对应的JobConf类实例，所以该参数不知道填什么……hadoop是3.3.0版本，很多以前的方法都被弃用了（版本太新了也不好），上网也没查到解决方案，所以换另一种实现方式，采用多个job顺序执行的方式（代价是增加很多IO操作，效率不高）。

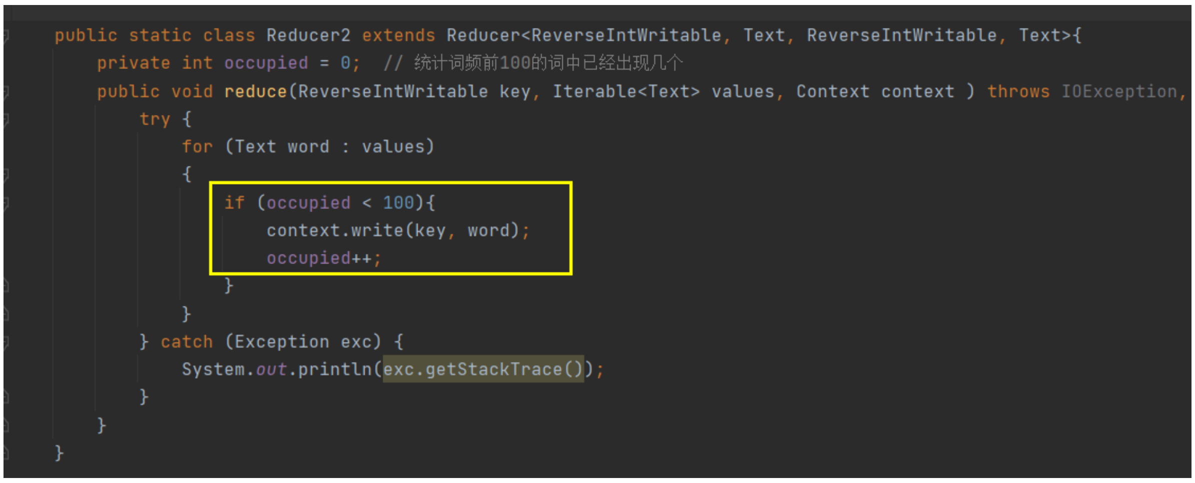
在job1中用MapReduce得出所有词的词频统计（乱序），但是reduce输出结果是以<词频，单词>的形式而非<单词，词频>的形式。

在job2中读取job1的输出结果，此时用map读取时仍将<词频，单词>作为输出结果，而在shuffle过程中hadoop会自动根据key进行排序，此处的key是词频，因此就不用自己再设计词频排序算法了。值得注意的是默认情况下由于IntWritable类的比较方式，词频会按从小到大的顺序进行排序，所以为了从大到小进行排序，需要自定义key类并自写CompareTo函数使得实际更大的数在比较时更“小”：



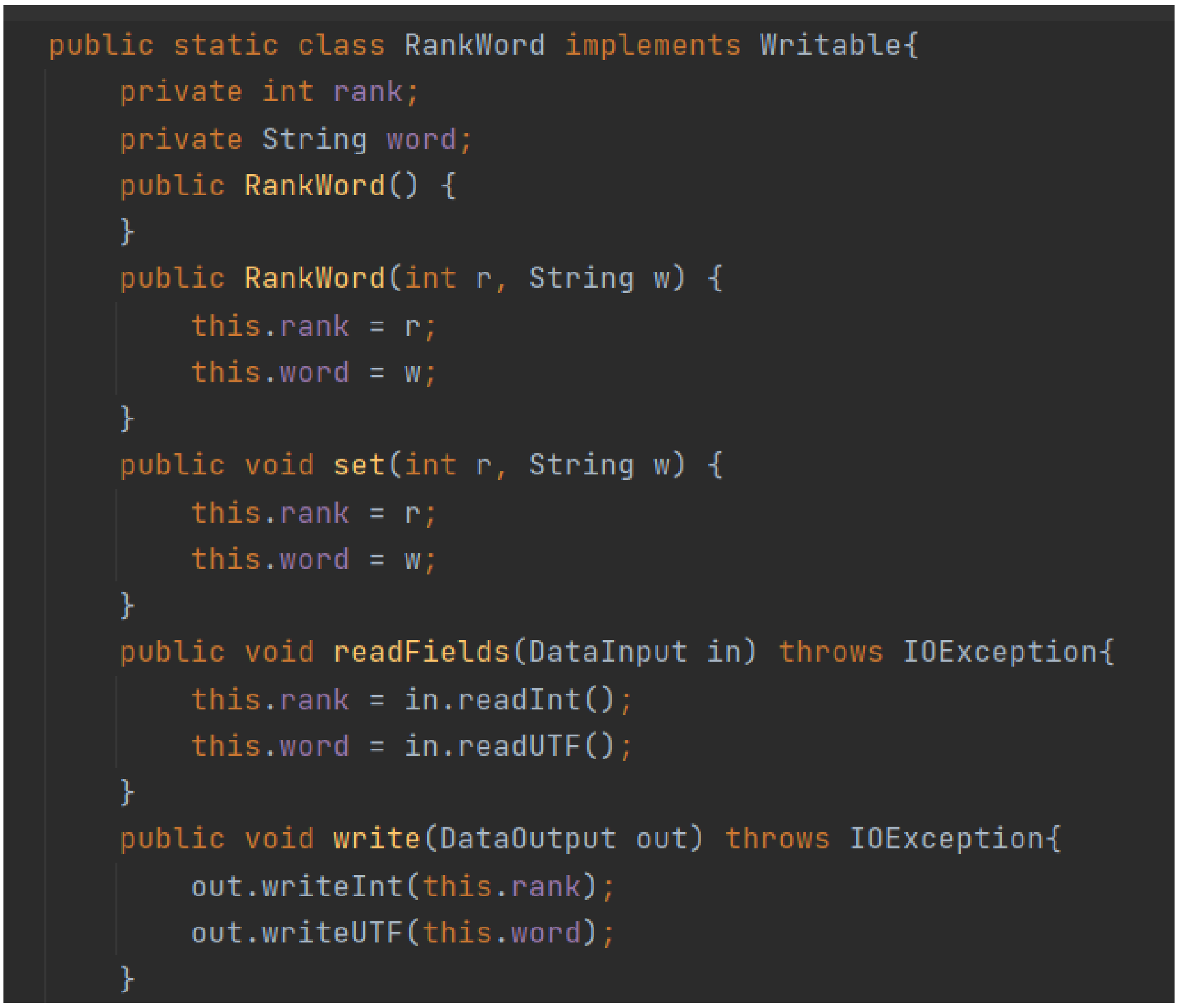
```java
public static class ReverseIntWritable implements WritableComparable<ReverseIntWritable>{
    private int value;

    public ReverseIntWritable() {}

    public ReverseIntWritable(int value) { this.set(value); }

    public void set(int value) { this.value = value; }

    public int get() { return this.value; }

    public void readFields(DataInput in) throws IOException {...}

    public void write(DataOutput out) throws IOException {...}

    public boolean equals(Object o) {...}

    public int hashCode() { return this.value; }

    public int compareTo(ReverseIntWritable o) {
        int thisValue = this.value;
        int thatValue = o.value;
        return thisValue > thatValue ? -1 : (thisValue == thatValue ? 0 : 1);
    }
```

而只显示前100个则是在Reduce类中定义变量occupied来进行计数，当计数超过100时就不再往context中写入数据：



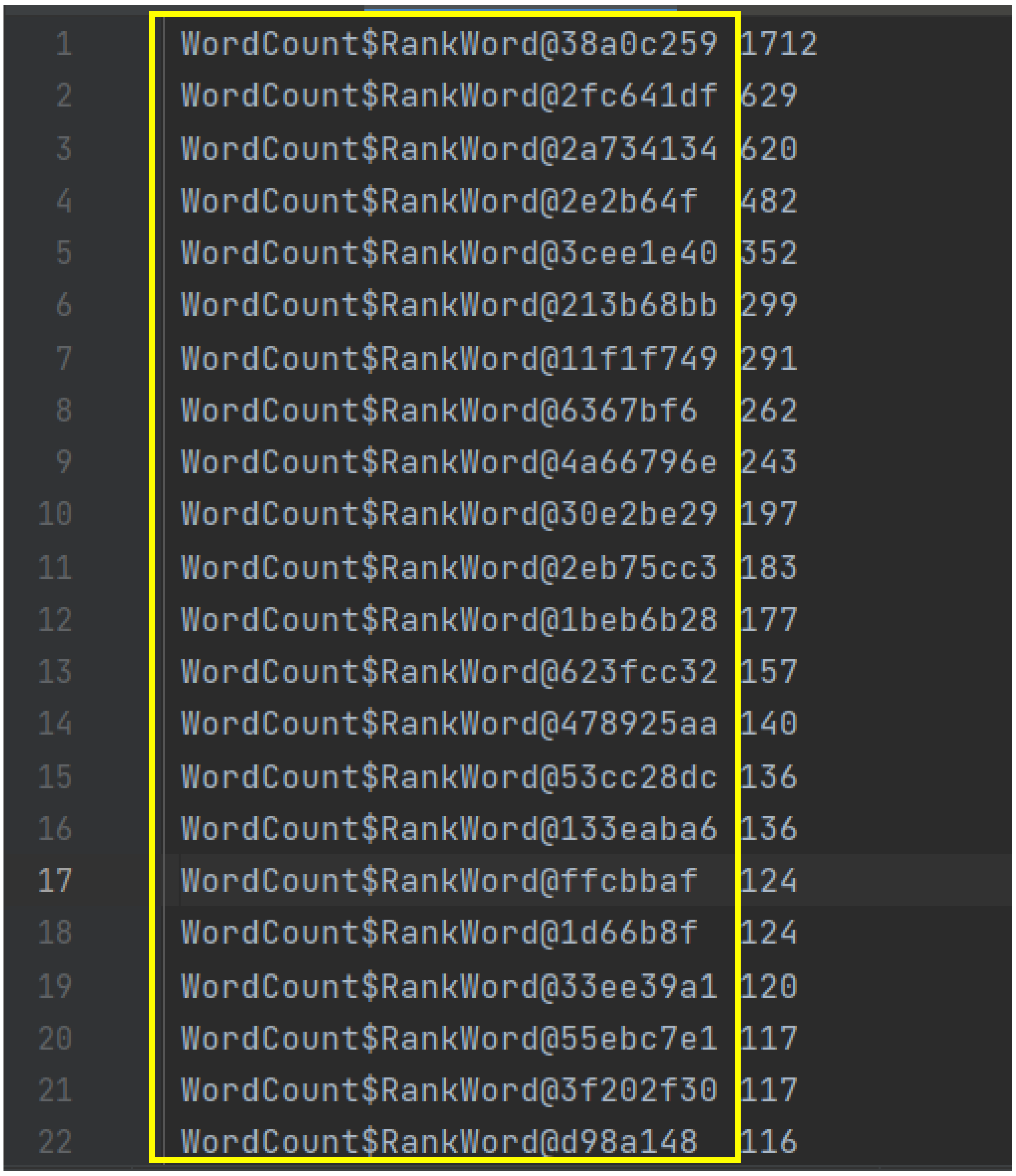
```java
public static class Reducer2 extends Reducer<ReverseIntWritable, Text, ReverseIntWritable, Text>{
    private int occupied = 0;  // 统计词频前100的词中已经出现几个
    public void reduce(ReverseIntWritable key, Iterable<Text> values, Context context ) throws IOException,
        try {
            for (Text word : values)
            {
                if (occupied < 100){
                    context.write(key, word);
                    occupied++;
                }
            }
        } catch (Exception exc) {
            System.out.println(exc.getStackTrace());
        }
    }
}
```

为了使输出符合 “排名：单词 词频”的格式，我自定义了一个RankWord类来实现Writable接口，意图是想输出RankWord类实例并使得每个RankWord类的实例都对应“排名+单词”的形式。上网查阅资料后，结合自身理解，我重写了write方法和readFields方法：



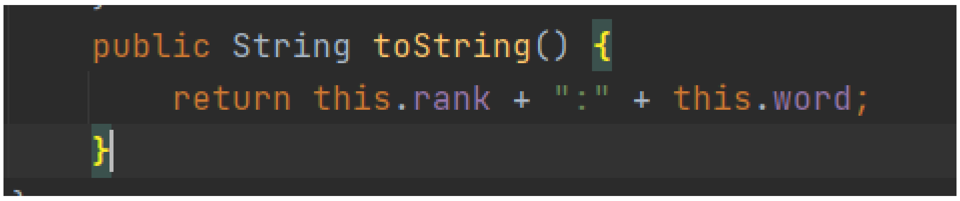
```java
public static class RankWord implements Writable{
    private int rank;
    private String word;
    public RankWord() {
    }
    public RankWord(int r, String w) {
        this.rank = r;
        this.word = w;
    }
    public void set(int r, String w) {
        this.rank = r;
        this.word = w;
    }
    public void readFields(DataInput in) throws IOException{
        this.rank = in.readInt();
        this.word = in.readUTF();
    }
    public void write(DataOutput out) throws IOException{
        out.writeInt(this.rank);
        out.writeUTF(this.word);
    }
```

但是此时运行程序，得到的结果却不符合预期，词频能正常显示，而RankWord类的实例却没有正常写入：



```
WordCount$RankWord@38a0c259	1712
WordCount$RankWord@2fc641df	629
WordCount$RankWord@2a734134	620
WordCount$RankWord@2e2b64f	482
WordCount$RankWord@3cee1e40	352
WordCount$RankWord@213b68bb	299
WordCount$RankWord@11f1f749	291
WordCount$RankWord@6367bf6	262
WordCount$RankWord@4a66796e	243
WordCount$RankWord@30e2be29	197
WordCount$RankWord@2eb75cc3	183
WordCount$RankWord@1beb6b28	177
WordCount$RankWord@623fcc32	157
WordCount$RankWord@478925aa	140
WordCount$RankWord@53cc28dc	136
WordCount$RankWord@133eaba6	136
WordCount$RankWord@ffcbbaf	124
WordCount$RankWord@1d66b8f	124
WordCount$RankWord@33ee39a1	120
WordCount$RankWord@55ebc7e1	117
WordCount$RankWord@3f202f30	117
WordCount$RankWord@d98a148	116
```

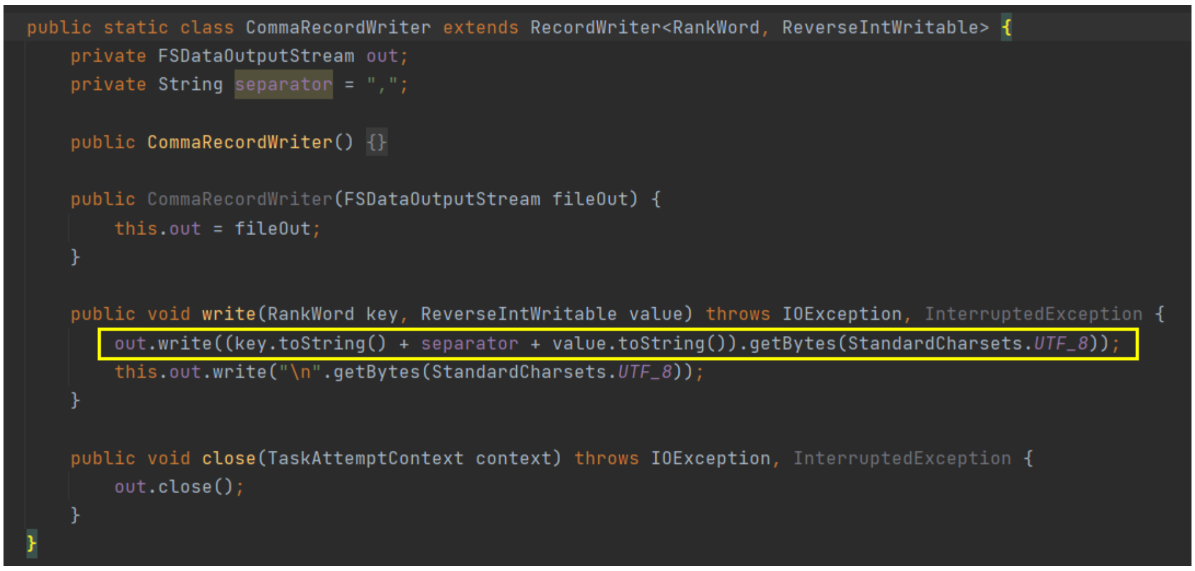
上网搜索也找不到相关资料，经过自身反复不断地摸索，费了很大的功夫，最终才发现是因为还需要实现一个toString方法！



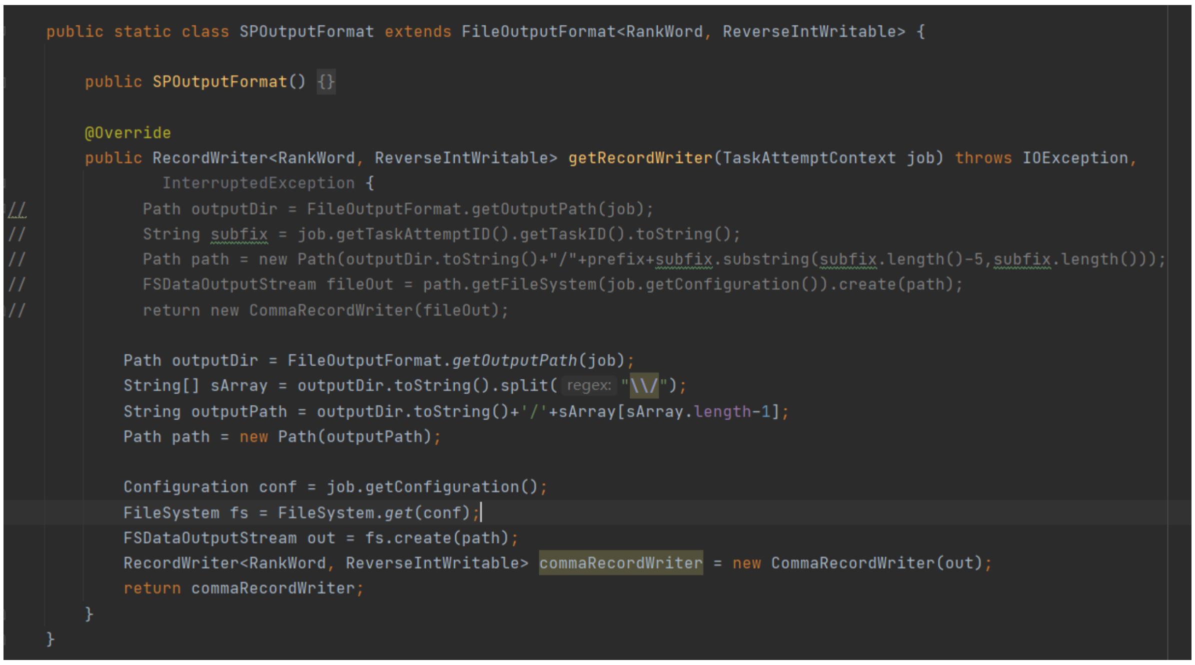
```java
public String toString() {
    return this.rank + ":" + this.word;
}
```

实现这个方法后终于可以正常显示了，但是本人还是有所不理解：按常理来说，写入文件所遵循的格式不是应该由write方法定义吗？如果由toString方法定义的话，那write方法起的作用又是什么呢？上网查阅之后略得知涉及序列化和反序列化，但还是不甚理解。总之，write这个名字实在太具有误导性，搞得我想破了头也想不出来为什么输出不遵循write方法定义的格式……

至于自定义输出文件名和分隔符采用逗号，则自定义RecordWriter和OutputFormat即可：



```java
public static class CommaRecordWriter extends RecordWriter<RankWord, ReverseIntWritable> {
    private FSDataOutputStream out;
    private String separator = ",";

    public CommaRecordWriter() {}

    public CommaRecordWriter(FSDataOutputStream fileOut) {
        this.out = fileOut;
    }

    public void write(RankWord key, ReverseIntWritable value) throws IOException, InterruptedException {
        out.write((key.toString() + separator + value.toString()).getBytes(StandardCharsets.UTF_8));
        this.out.write("\n".getBytes(StandardCharsets.UTF_8));
    }

    public void close(TaskAttemptContext context) throws IOException, InterruptedException {
        out.close();
    }
}
```



```java
    public static class SPOutputFormat extends FileOutputFormat<RankWord, ReverseIntWritable> {

        public SPOutputFormat() {}

        @Override
        public RecordWriter<RankWord, ReverseIntWritable> getRecordWriter(TaskAttemptContext job) throws IOException,
                InterruptedException {
//            Path outputDir = FileOutputFormat.getOutputPath(job);
//            String subfix = job.getTaskAttemptID().getTaskID().toString();
//            Path path = new Path(outputDir.toString()+"/"+prefix+subfix.substring(subfix.length()-5,subfix.length()));
//            FSDataOutputStream fileOut = path.getFileSystem(job.getConfiguration()).create(path);
//            return new CommaRecordWriter(fileOut);

            Path outputDir = FileOutputFormat.getOutputPath(job);
            String[] sArray = outputDir.toString().split( regex: "\\/");
            String outputPath = outputDir.toString()+'/'+sArray[sArray.length-1];
            Path path = new Path(outputPath);

            Configuration conf = job.getConfiguration();
            FileSystem fs = FileSystem.get(conf);
            FSDataOutputStream out = fs.create(path);
            RecordWriter<RankWord, ReverseIntWritable> commaRecordWriter = new CommaRecordWriter(out);
            return commaRecordWriter;
        }
    }
```

关于忽略标点和停词，由于逻辑不同（忽略标点只需直接调用replace方法全部替换成空字符即可，但是停词则不能盲目替换，比如在your中将停词you替换成空字符后就变成了r，这是不合规的），所以我将停词文件和标点文件分开处理，对于标点直接用replace全部替换成空，而对于去除标点后的文本，提取出每一个单词后，判断其是否在停词列表中（同时判断其是否是数字且是否长度小于3），若否，则调用context.write写入，若是，则不写入。

以上部分可以实现对于单个txt文本进行统计。为了能自动对所有txt文本都执行该统计，我的实现方式是编写for循环，同一时刻只有一个txt文本被处理，当一个文本被处理完毕后对下一个文本执行同样的操作。

而至于对所有txt文件进行词频总排行，其实和单个文件比较类似，主要的区别就在于输入文件是多个，而非一个，此外还需要对Mapper和Reducer进行一些小的修改以使得输出输入格式一致。