JAVA 编程进阶上机报告

学　院　智能与计算学部

专　业　软件工程

班　级　6 班

学　号　3018216298

姓　名　米思成

一、实验要求

编写程序，统计了不起的盖茨比中各个单词出现的频次。

注意事项

1. 尝试使用不同的 stream 进行读文件操作。
2. 异常处理（例如文件不存在，文件没有读权限，文件编码错误等）

输入:

了不起的盖茨比（英文版）.txt

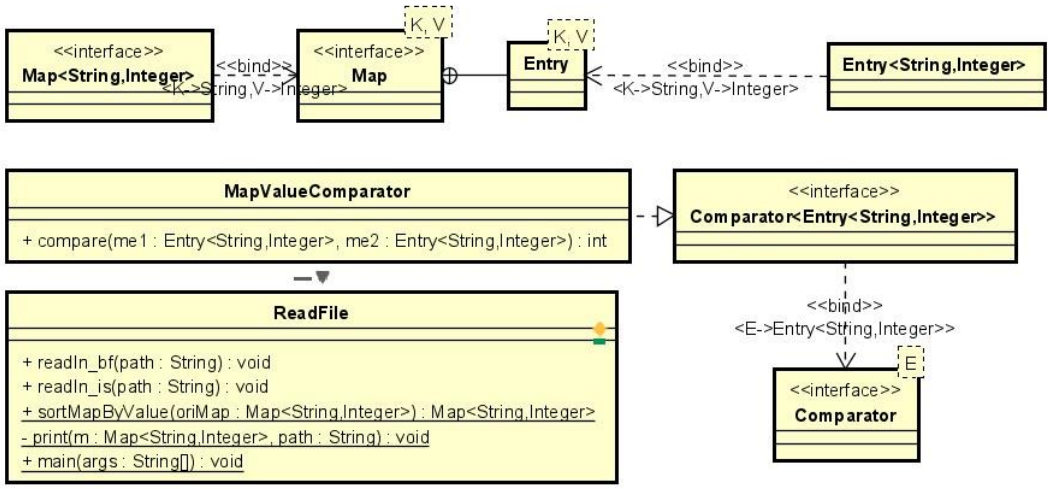
(其中一个)

输出:

为输入文件，创建一个 output.txt

输出格式：单词+空格+频次，结果按照单词的频次倒序排列

二、UML 图（设计）



设计思路：

相同部分：

sortMapByValue（）方法用于将存入 map 的单词按照词频降序排列。

print（）方法用于将排序好的 map 按照要求格式输出。

main（）用于开始实验。

不同部分：

readIn_bf（）方法用于通过 BufferedReader 读入文件，并进行数据处理和词频统计。

readIn_is（）方法用于通过 InputStream 读入文件，并进行数据处理和词频统计。

三、源代码

```
public class ReadFile {

    //通过BufferedReader读入文件
    public void readIn_bf(String path) throws IOException {

        File file = new File(path);
        BufferedReader br = null;
        Map<String,Integer> map = new TreeMap<String,Integer>();
        try
        {
            br=new BufferedReader(new FileReader(file));
        }
        catch (FileNotFoundException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        String line="";
        int cnt=0;
        List<String> l = new ArrayList<String>();
        try
        {
             while ((line = br.readLine()) != null)
             {
               //数组里面每个元素是一个单词
               String str[]=line.split(" ");

               //数据处理
```

```
                //将所有的大写转为小写并除去所有符号
                for(int i=0;i<str.length;i++) {
                    str[i] = str[i].toLowerCase();
                    String
regEx="[\n`~!@#$%^&*()+=|{}':;',\\[\\].<>/?~！@#￥%……&*（ ）——+|{}
【】‘；：”“’。，、？]";
                    String x = "";
                    str[i] = str[i].replaceAll(regEx,x);
                    l.add(str[i]);
                }
            }
//              System.out.println(l.size());


            //统计词频
            ii:
              for(int i=0; i<l.size(); i++) {


                //判断之前是否出现过这个字符串
                for(int k=0;k<i;k++) {
                    if(l.get(i).equals(l.get(k)))
                        continue ii;
                }


                //count此字符串出现过的次数
                int temp = 1;
                for(int j=i+1; j<l.size(); j++) {

                    if(!l.get(i).equals(l.get(j))) {

                    }
                    else {
                        temp++;
                    }
                }
                //把字符串添加入Map
                map.put((String) l.get(i), temp);
              }
```

```
            Map<String, Integer> resultMap = sortMapByValue(map);
//按Key进行排序
            print(resultMap,"BufferedReader_output.txt");

        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    //通过InputStream读入文件
    public void readIn_is(String path) throws IOException {

        File file = new File(path);
        Map<String,Integer> map = new TreeMap<String,Integer>();
        List<String> l = new ArrayList<String>();
        InputStream is = new FileInputStream(file);
        StringBuffer sb = null;
        try {
            sb = new StringBuffer();
            int temp = 0;
            do {
                temp = is.read();
                if(temp!=-1) {
                    sb.append((char)temp);
                }
            }while(temp!=-1);
        }catch(IOException e){
            e.printStackTrace();
        }finally{
            is.close();
        }

        String str[]=sb.toString().split(" ");
//      System.out.println(str.length);

        //数据处理
```

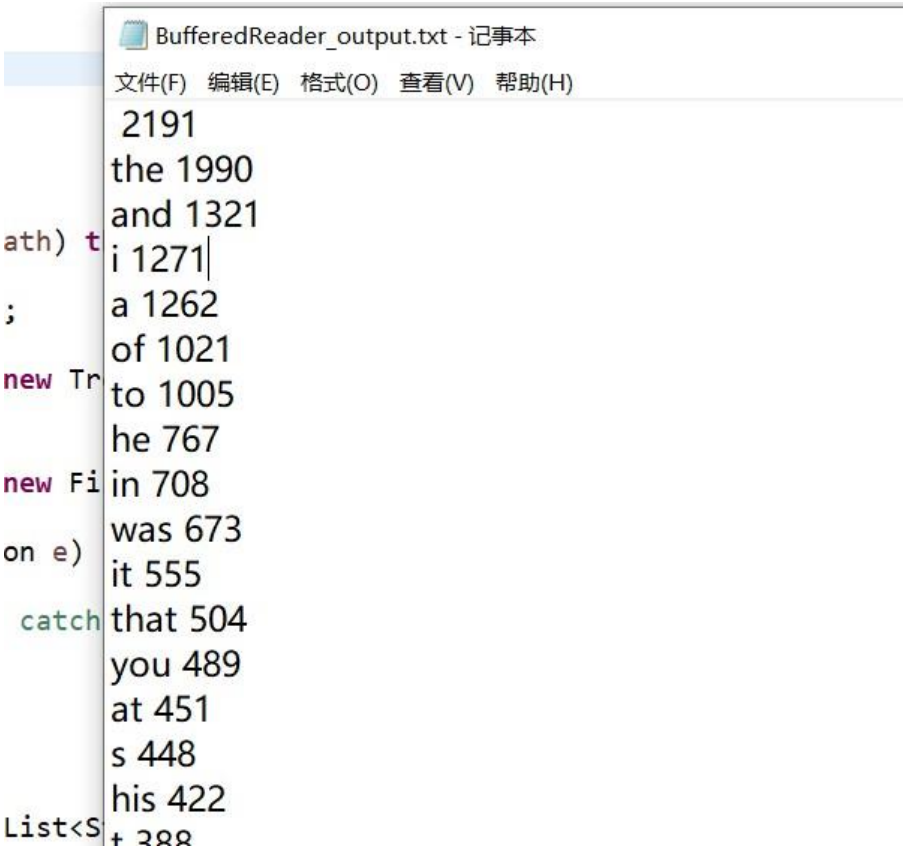
```
        //将所有的大写转为小写并除去所有符号
        for(int i=0;i<str.length;i++) {
            str[i] = str[i].toLowerCase();
            String regEx="[\n`~!@#$%^&*()+=|{}':;',\\[\\].<>/?~！
@#￥%……&*（）——+|{}【】‘；：”“’。，、？]";
            String x = "";
            str[i] = str[i].replaceAll(regEx,x);
            l.add(str[i]);
        }

//    System.out.println(l.size());

    //统计词频
    ii:
        for(int i=0; i<l.size(); i++) {

            //判断之前是否出现过这个字符串
            for(int k=0;k<i;k++) {
                if(l.get(i).equals(l.get(k)))
                    continue ii;
            }

            //count此字符串出现过的次数
            int temp = 1;
            for(int j=i+1; j<l.size(); j++) {

                if(!l.get(i).equals(l.get(j))) {

                }
                else {
                    temp++;
                }
            }
            //把字符串添加入Map
            map.put((String) l.get(i), temp);
        }
```

```
    Map<String, Integer> resultMap = sortMapByValue(map);    //
按Key进行排序
    print(resultMap,"InputStream_output.txt");


    }


    //排序
    public static Map<String, Integer> sortMapByValue(Map<String,
Integer> oriMap) {
        if (oriMap == null || oriMap.isEmpty()) {
            return null;
        }
        Map<String, Integer> sortedMap = new LinkedHashMap<String,
Integer>();
        List<Map.Entry<String, Integer>> entryList = new
ArrayList<Map.Entry<String, Integer>>(
                oriMap.entrySet());
        Collections.sort(entryList, new MapValueComparator());

        Iterator<Map.Entry<String, Integer>> iter =
entryList.iterator();
        Map.Entry<String, Integer> tmpEntry = null;
        while (iter.hasNext()) {
            tmpEntry = iter.next();
            sortedMap.put(tmpEntry.getKey(),
tmpEntry.getValue());
        }
        return sortedMap;
    }


    //输出结果
    private static void print(Map<String, Integer> m, String path)
{
       Set<Map.Entry<String,Integer>> set = m.entrySet();
       Iterator<Map.Entry<String,Integer>> iter = set.iterator();
       File file = new File("E:/"+path);
        try {
         file.createNewFile();
```

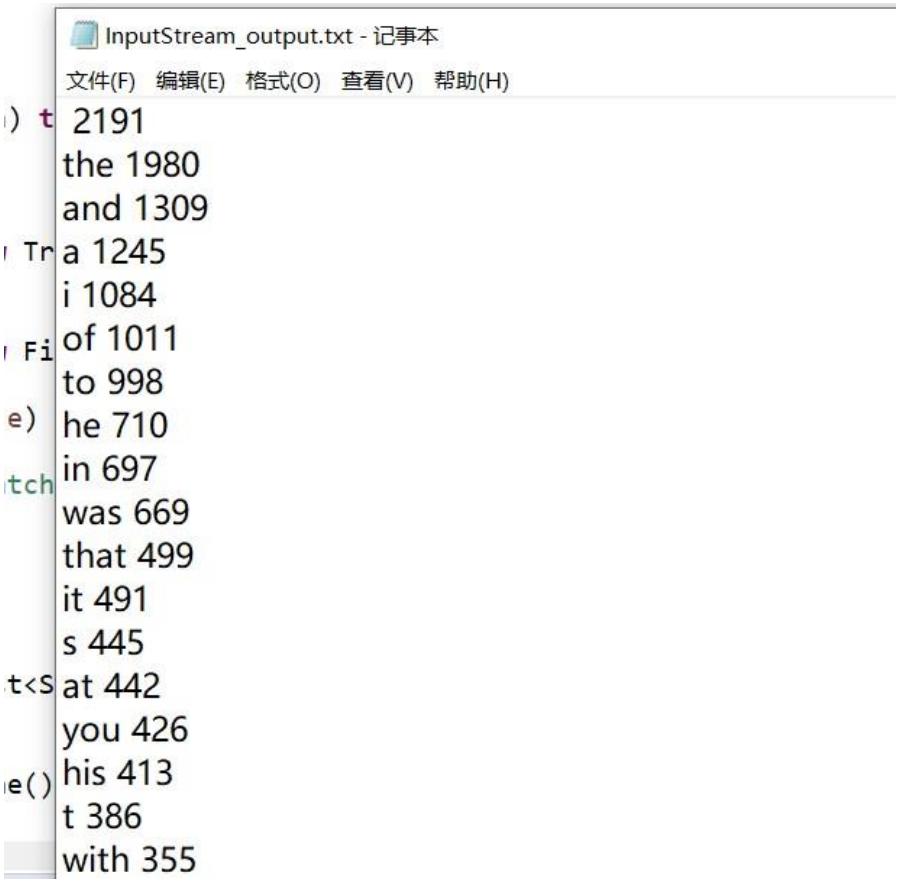
```
        FileWriter fileWriter = new
FileWriter(file.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fileWriter);
         while(iter.hasNext()) {
           Map.Entry<String, Integer> ele = iter.next();
           String temp = ele.getKey()+" "+ele.getValue()+"\n";
           bw.write(temp);
       }
           bw.close();
         }catch(IOException e){
           e.printStackTrace();
         }

    }


    public static void main(String[] args) throws IOException
    {
        ReadFile L=new ReadFile();

        String path="E:/了不起的盖茨比英文.txt";

        L.readIn_bf(path);
        L.readIn_is(path);
    }

}

class MapValueComparator implements Comparator<Map.Entry<String,
Integer>> {

    @Override
    public int compare(Entry<String, Integer> me1, Entry<String,
Integer> me2) {

        return -me1.getValue().compareTo(me2.getValue());
    }
}
```

四、实验结果

1. 通过 BufferedReader 读入：



2. 通过 InputStream 读入：



五、心得体会

这次试验的要求看起来容易，但实际操作时候还是遇到了一些坑，比如忘记关闭输入流，忘记进行 IO 操作时抛异常……其中最大的问题莫过于按照 value 值对 map 进行降序排列，实现之前查找了很多资料，今后还是要多多注意细节问题。