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	3018216281

姓　名 ＿＿＿朱明煊＿＿＿＿＿＿

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)

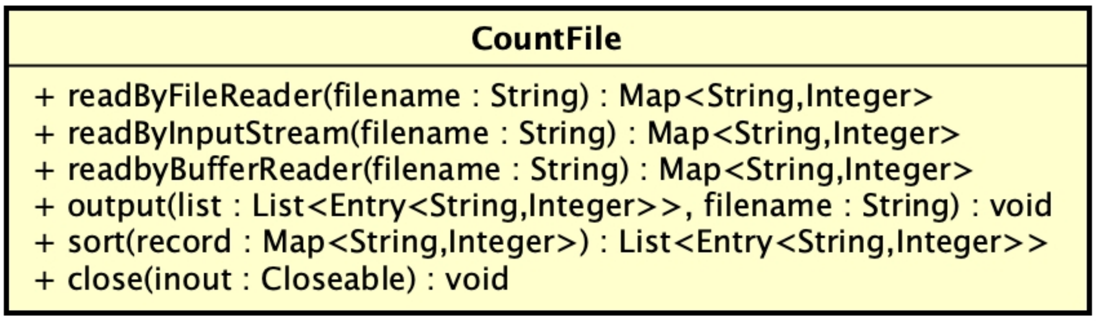
输出:

为输入文件，创建一个 output.txt

输出格式如下，单词+空格+频次，结果按照单词的频次倒序排列

```
hello 123

hi 12

i 1
```

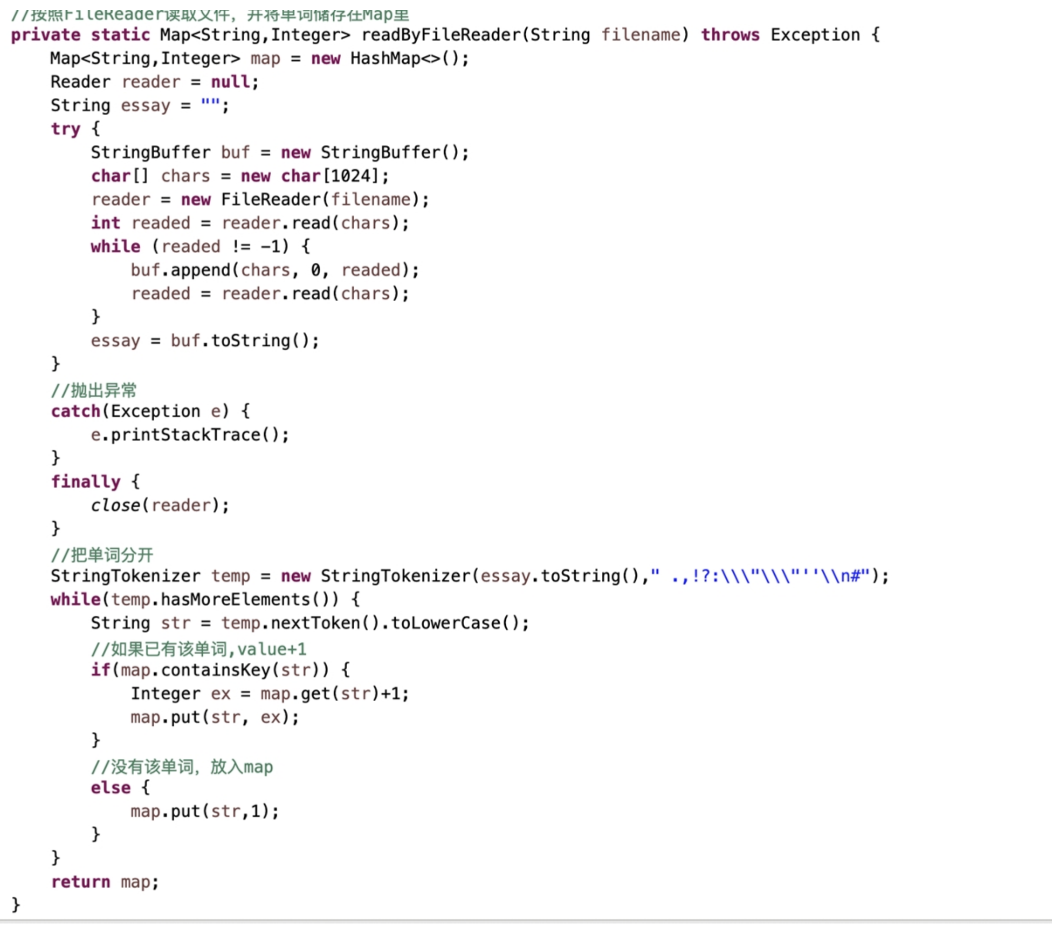
二、设计思想与类图解析


CountFile
+ readByFileReader(filename : String) : Map<String,Integer>
+ readByInputStream(filename : String) : Map<String,Integer>
+ readbyBufferReader(filename : String) : Map<String,Integer>
+ output(list : List<Entry<String,Integer>>, filename : String) : void
+ sort(record : Map<String,Integer>) : List<Entry<String,Integer>>
+ close(inout : Closeable) : void

思想：为了实现题目的要求，我们主要在类中实现如下方法：

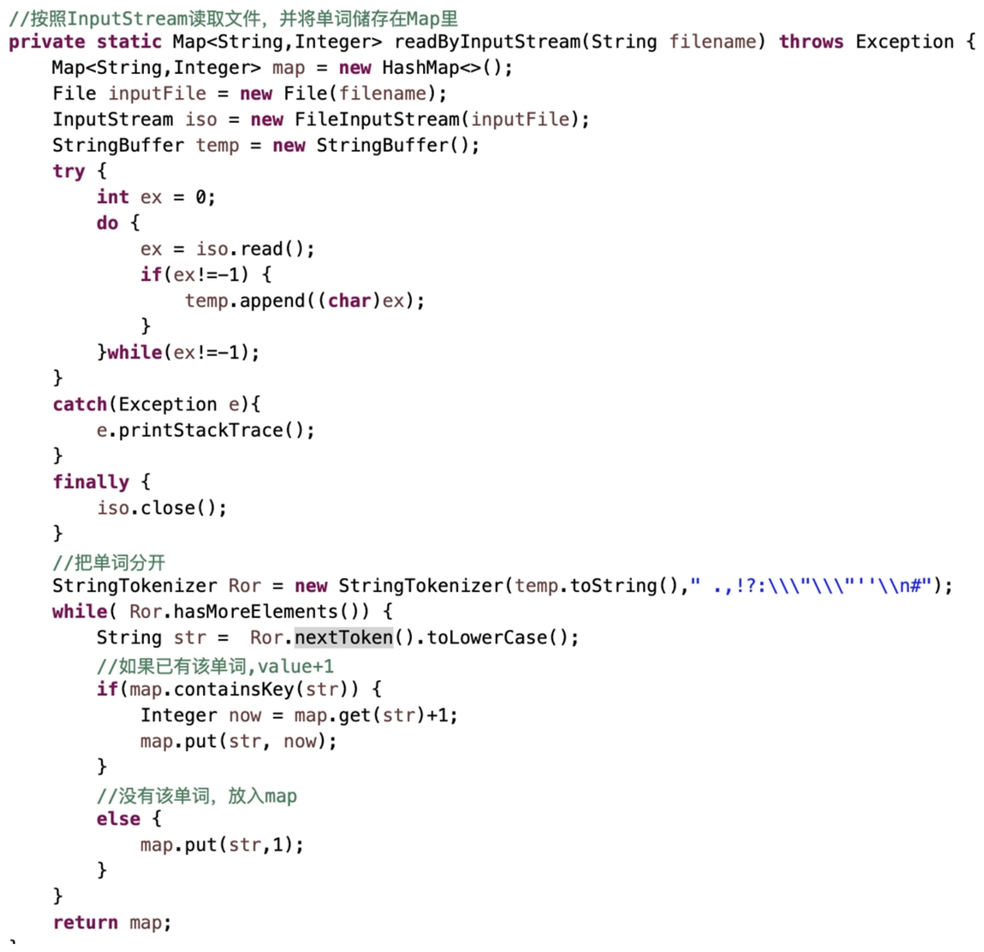
1、文件读取

2、储存每个单词的数量
3、将单词按照个数排序
4、将结果输出到新的文本中
同时在读入文件时要抛出异常。
首先使用 FileReader 方法读入文件，并同时进行异常处理，这里的 Exception 类能捕获全部异常。在读取内容后，我们使用 StringTokenizer 方法
将单词分别读取出来，存入到 Map 中，因为 Map 的 Key 不重复原则，我们可以存取每一个单词与单词的数量。



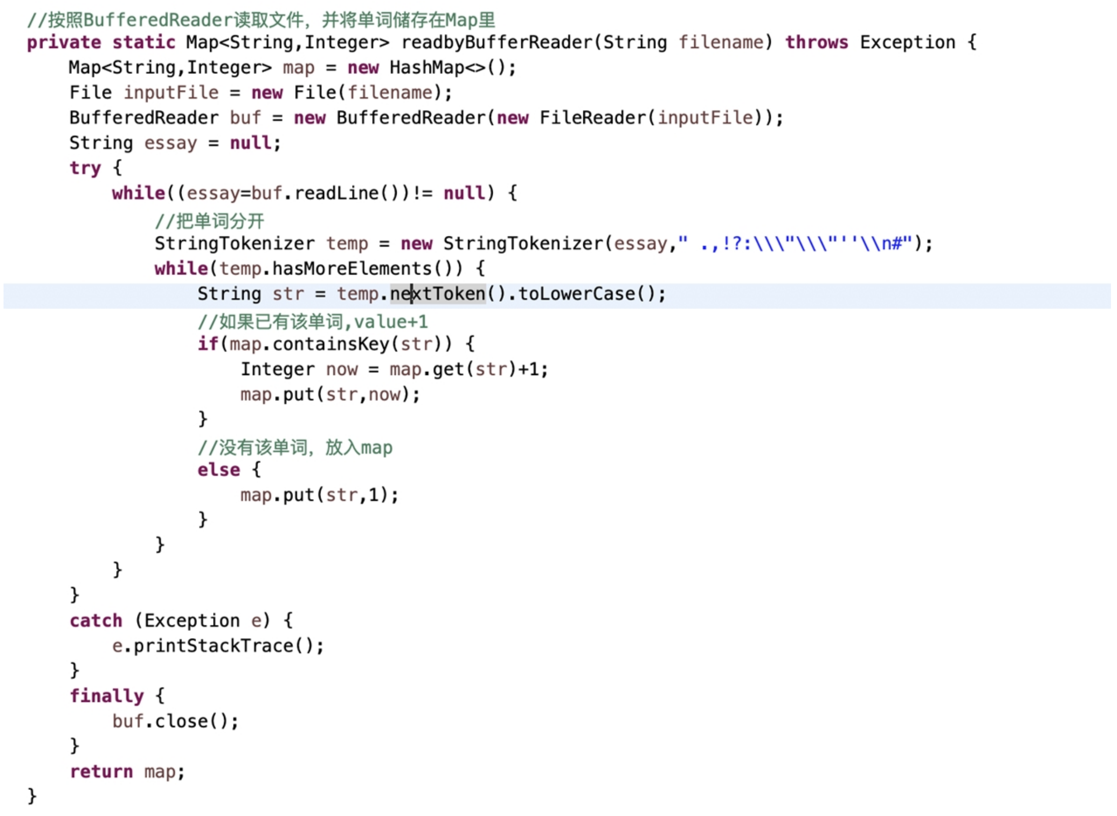
```java
//按照FileReader读取文件，并将单词储存在Map里
private static Map<String,Integer> readByFileReader(String filename) throws Exception {
    Map<String,Integer> map = new HashMap<>();
    Reader reader = null;
    String essay = "";
    try {
        StringBuffer buf = new StringBuffer();
        char[] chars = new char[1024];
        reader = new FileReader(filename);
        int readed = reader.read(chars);
        while (readed != -1) {
            buf.append(chars, 0, readed);
            readed = reader.read(chars);
        }
        essay = buf.toString();
    }
    //抛出异常
    catch(Exception e) {
        e.printStackTrace();
    }
    finally {
        close(reader);
    }
    //把单词分开
    StringTokenizer temp = new StringTokenizer(essay.toString()," .,!?:\\\"\\\"''\\n#");
    while(temp.hasMoreElements()) {
        String str = temp.nextToken().toLowerCase();
        //如果已有该单词,value+1
        if(map.containsKey(str)) {
            Integer ex = map.get(str)+1;
            map.put(str, ex);
        }
        //没有该单词，放入map
        else {
            map.put(str,1);
        }
    }
    return map;
}
```

之后我们使用了 InputStream 进行数据读取，同样进行异常处理与 Map 储存。



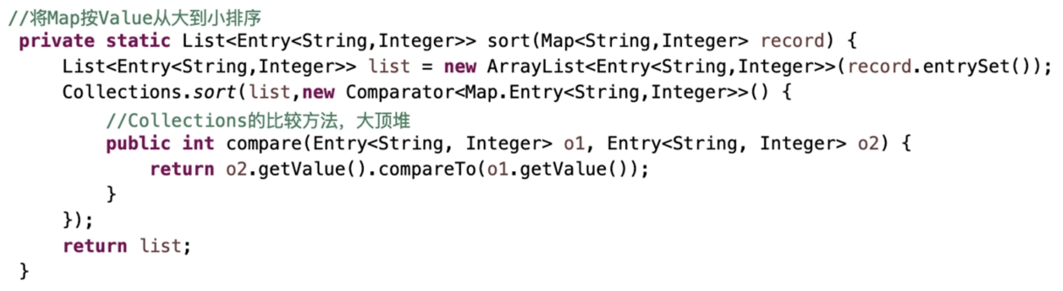
```java
//按照InputStream读取文件，并将单词储存在Map里
private static Map<String,Integer> readByInputStream(String filename) throws Exception {
    Map<String,Integer> map = new HashMap<>();
    File inputFile = new File(filename);
    InputStream iso = new FileInputStream(inputFile);
    StringBuffer temp = new StringBuffer();
    try {
        int ex = 0;
        do {
            ex = iso.read();
            if(ex!=-1) {
                temp.append((char)ex);
            }
        }while(ex!=-1);
    }
    catch(Exception e){
        e.printStackTrace();
    }
    finally {
        iso.close();
    }
    //把单词分开
    StringTokenizer Ror = new StringTokenizer(temp.toString()," .,!?:\\\"\\\"''\\n#");
    while( Ror.hasMoreElements()) {
        String str =  Ror.nextToken().toLowerCase();
        //如果已有该单词,value+1
        if(map.containsKey(str)) {
            Integer now = map.get(str)+1;
            map.put(str, now);
        }
        //没有该单词，放入map
        else {
            map.put(str,1);
        }
    }
    return map;
```

也使用了 BufferReader 进行数据读取，同样进行异常处理与 Map 储存。



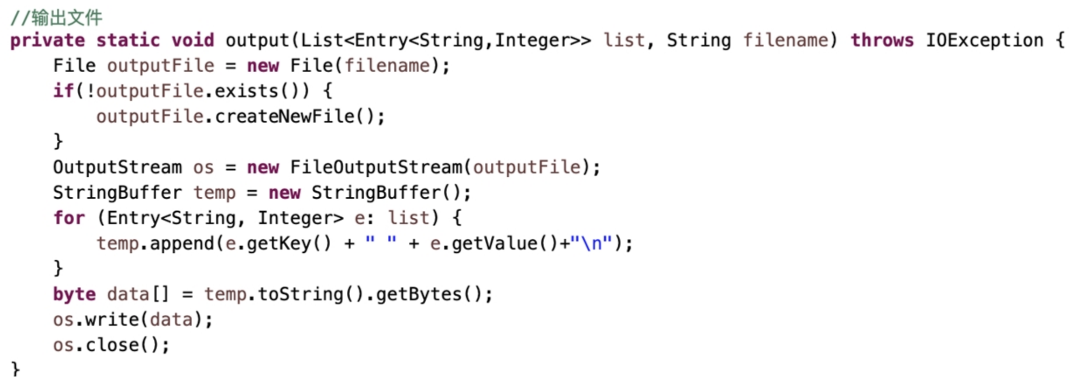
```java
//按照BufferedReader读取文件，并将单词储存在Map里
private static Map<String,Integer> readbyBufferReader(String filename) throws Exception {
    Map<String,Integer> map = new HashMap<>();
    File inputFile = new File(filename);
    BufferedReader buf = new BufferedReader(new FileReader(inputFile));
    String essay = null;
    try {
        while((essay=buf.readLine())!= null) {
            //把单词分开
            StringTokenizer temp = new StringTokenizer(essay," .,!?:\\\"\\\"''\\n#");
            while(temp.hasMoreElements()) {
                String str = temp.nextToken().toLowerCase();
                //如果已有该单词,value+1
                if(map.containsKey(str)) {
                    Integer now = map.get(str)+1;
                    map.put(str,now);
                }
                //没有该单词，放入map
                else {
                    map.put(str,1);
                }
            }
        }
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    finally {
        buf.close();
    }
    return map;
}
```

下一步我们将 Map 按照 value 大小进行排序，并存到一个列表之中，这里我们使用了 Collection 特有的排序函数



```java
//将Map按Value从大到小排序
private static List<Entry<String,Integer>> sort(Map<String,Integer> record) {
    List<Entry<String,Integer>> list = new ArrayList<Entry<String,Integer>>(record.entrySet());
    Collections.sort(list,new Comparator<Map.Entry<String,Integer>>() {
        //Collections的比较方法，大顶堆
        public int compare(Entry<String, Integer> o1, Entry<String, Integer> o2) {
            return o2.getValue().compareTo(o1.getValue());
        }
    });
    return list;
}
```

将结果进行输出，把列表按格式输入到文件中，采取 OutputStream 流



```java
//输出文件
private static void output(List<Entry<String,Integer>> list, String filename) throws IOException {
    File outputFile = new File(filename);
    if(!outputFile.exists()) {
        outputFile.createNewFile();
    }
    OutputStream os = new FileOutputStream(outputFile);
    StringBuffer temp = new StringBuffer();
    for (Entry<String, Integer> e: list) {
        temp.append(e.getKey() + " " + e.getValue()+"\n");
    }
    byte data[] = temp.toString().getBytes();
    os.write(data);
    os.close();
}
```

三、源代码

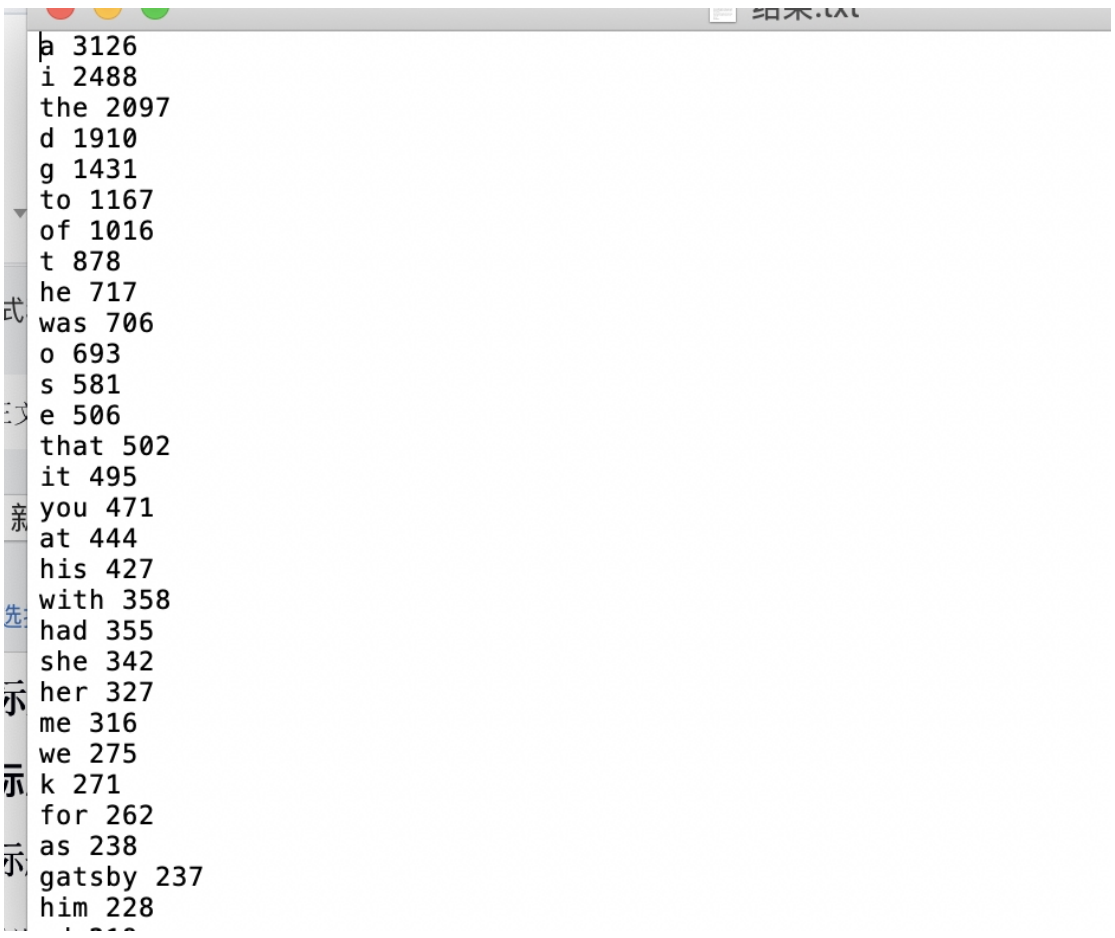
```
package Lab2;

import java.io.BufferedReader;

import java.io.Closeable;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.OutputStream;

import java.io.Reader;

import java.io.InputStream;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.StringTokenizer;

import java.io.File;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;


public class CountFile {


    //将Map按Value从大到小排序

     private            static            List<Entry<String,Integer>>
sort(Map<String,Integer> record) {
        List<Entry<String,Integer>>        list        =        new
ArrayList<Entry<String,Integer>>(record.entrySet());

        Collections.sort(list,new
Comparator<Map.Entry<String,Integer>>() {

            //Collections的比较方法，大顶堆

            public    int    compare(Entry<String,    Integer>    o1,
Entry<String, Integer> o2) {

                return o2.getValue().compareTo(o1.getValue());

            }

        });

        return list;

    }


    //按照FileReader读取文件，并将单词储存在Map里

    private  static  Map<String,Integer>  readByFileReader(String
filename) throws Exception {

        Map<String,Integer> map = new HashMap<>();

        Reader reader = null;

        String essay = "";

        try {

            StringBuffer buf = new StringBuffer();

            char[] chars = new char[1024];

            reader = new FileReader(filename);

            int readed = reader.read(chars);

            while (readed != -1) {

                buf.append(chars, 0, readed);
                readed = reader.read(chars);
            }
            essay = buf.toString();
        }
        //抛出异常
        catch(Exception e) {
            e.printStackTrace();
        }
        finally {
            close(reader);
        }
        //把单词分开
        StringTokenizer temp = new StringTokenizer(essay.toString(),″ .,!?:\\\″\\\″’’\\n#″);
        while(temp.hasMoreElements()) {
            String str = temp.nextToken().toLowerCase();
            //如果已有该单词,value+1
            if(map.containsKey(str)) {
                Integer ex = map.get(str)+1;
                map.put(str, ex);
            }
            //没有该单词，放入map
            else {
                map.put(str,1);
            }
        }
        return map;
    }

    //关闭流
    private static void close(Closeable inout) {
        if (inout != null) {
            try {
                inout.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    //按照 InputStream 读取文件，并将单词储存在 Map 里
    private  static  Map<String,Integer>  readByInputStream(String
filename) throws Exception {
        Map<String,Integer> map = new HashMap<>();
        File inputFile = new File(filename);
        InputStream iso = new FileInputStream(inputFile);
        StringBuffer temp = new StringBuffer();
        try {
            int ex = 0;
            do {
                ex = iso.read();
                if(ex!=-1) {
                    temp.append((char)ex);
                }
            }while(ex!=-1);
        }
        catch(Exception e){
            e.printStackTrace();
        }
        finally {
             iso.close();
        }
        //把单词分开
        StringTokenizer Ror = new StringTokenizer(temp.toString(),″
.,!?:\\\″\\\″’’\\n#″);
        while( Ror.hasMoreElements()) {
            String str =  Ror.nextToken().toLowerCase();
            //如果已有该单词,value+1
            if(map.containsKey(str)) {
                Integer now = map.get(str)+1;
                map.put(str, now);
            }
            //没有该单词，放入map
            else {
                map.put(str,1);
            }
        }
        return map;
    }

    //按照 BufferedReader 读取文件，并将单词储存在 Map 里
    private static Map<String,Integer> readbyBufferReader(String filename) throws Exception {
        Map<String,Integer> map = new HashMap<>();
        File inputFile = new File(filename);
        BufferedReader buf = new BufferedReader(new FileReader(inputFile));
        String essay = null;
        try {
            while((essay=buf.readLine())!= null) {
                //把单词分开
                StringTokenizer temp = new StringTokenizer(essay,″ .,!?:\\\″\\\″’’\\n#″);
                while(temp.hasMoreElements()) {
                    String str = temp.nextToken().toLowerCase();
                    //如果已有该单词,value+1
                    if(map.containsKey(str)) {
                        Integer now = map.get(str)+1;
                        map.put(str,now);
                    }
                    //没有该单词，放入 map
                    else {
                        map.put(str,1);
                    }
                }
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            buf.close();
        }
        return map;
    }

    //输出文件
    private static void output(List<Entry<String,Integer>> list,
String filename) throws IOException {
        File outputFile = new File(filename);
        if(!outputFile.exists()) {
            outputFile.createNewFile();
        }
        OutputStream os = new FileOutputStream(outputFile);
        StringBuffer temp = new StringBuffer();
        for (Entry<String, Integer> e: list) {
            temp.append(e.getKey() + " " + e.getValue()+"\n");
        }
        byte data[] = temp.toString().getBytes();
        os.write(data);
        os.close();
    }
    public static void main(String args[]) throws Exception {
        Map<String, Integer> record = new HashMap<>();
        Map<String, Integer> record2 = new HashMap<>();
        Map<String, Integer> record3 = new HashMap<>();
        String pass = "/Users/zhumingxuan/Desktop/了不起的盖茨比英
文.txt";
        String to = "/Users/zhumingxuan/Desktop/结果.txt";
        String to2 = "/Users/zhumingxuan/Desktop/结果2.txt";
        String to3 = "/Users/zhumingxuan/Desktop/结果3.txt";
        //用FileReader读入
        record =  readByFileReader(pass);
        List<Entry<String,Integer>> list = sort(record);
        output(list,to);
        //用InputStream读入
        record2 =  readByInputStream(pass);
        List<Entry<String,Integer>> list2 = sort(record2);
        output(list2,to2);
        //用BufferReader读入
        record3 =  readbyBufferReader(pass);
```

```
            List<Entry<String,Integer>> list3 = sort(record3);
            output(list3,to3);
        }
}
```

四、实验结果

三种方法输出了三个文件，三个文件结果相同（放在文件夹中）

部分结果



```
a 3126
i 2488
the 2097
d 1910
g 1431
to 1167
of 1016
t 878
he 717
was 706
o 693
s 581
e 506
that 502
it 495
you 471
at 444
his 427
with 358
had 355
she 342
her 327
me 316
we 275
k 271
for 262
as 238
gatsby 237
him 228
```