

第二次实验---词频统计器

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

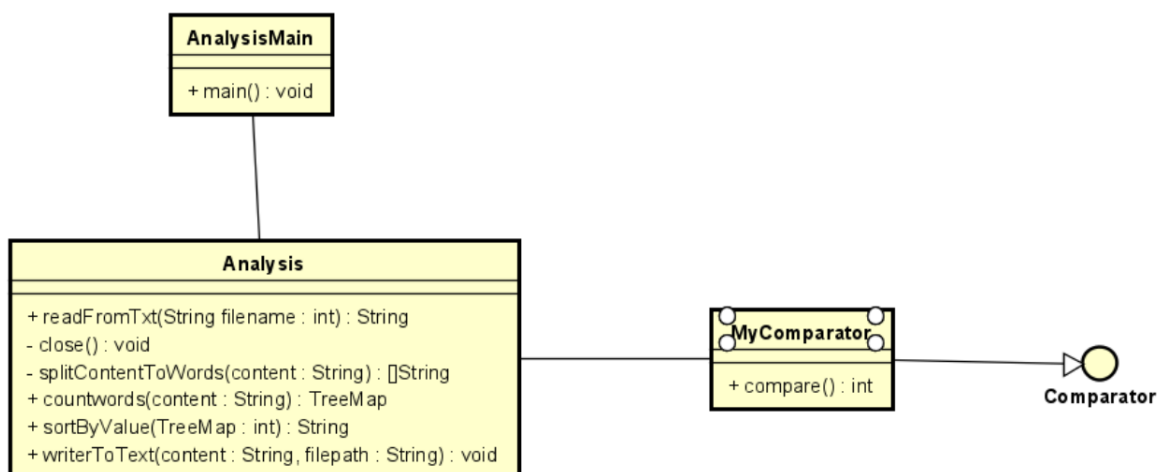
输出格式如下，单词+空格+频次，结果按照单词的频次倒序排列

hello 123

hi 12

i 1

实验思路和UML图



实验思路：

实验有三个文件，main、一个词频统计器功能的文件、还有一个比较排序的类实现了comparator

main函数通过构造函数调用词频统计器。因为词频统计器只有一个功能。

实验代码

```

package container_IO;

import java.io.File;

public class AnalysisMain {

    public static void main(String[] arg) throws Exception {
        String origin_filepath = "D:/temp/了不起的盖茨比英文.txt";
        String final_filepath = "D:" + File.separator +
"temp"+File.separator+"output.txt";
        Analysis analysis = new Analysis(origin_filepath,final_filepath);
    }
}

```

```

package container_IO;

import java.io.BufferedReader;
import java.io.Closeable;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map.Entry;
import java.util.TreeMap;

public class Analysis {

    public Analysis(String origin_filepath ,String final_filepath) throws
Exception {
        //读取原始的内容到字符串
        String origin_content =this.readFromTxt(origin_filepath);
        //处理得到的字符串为NewContent
        String final_Content = this.sortByValue(countwords(origin_content));
        //写文件
        this.writeToText(final_Content,final_filepath);
    }

    /**从文章中读内容需要在每一行的前面加空格**/
    public String readFromTxt(String filename) throws Exception {
        BufferedReader buf = null;
        try{
            File file = new File(filename);
            // 使用文件输入流实例化BufferedReader类对象
            buf = new BufferedReader(new FileReader(file));
            String str = null; // 接收输入数据
            String content = null;//目标字符串
            while ((str = buf.readLine()) != null) { // 读取数据并判断是否存在
                content = content + " "+str; // 输出读取内容
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return content;
    }

    //统计单词出现次数
    public Map<String, Integer> countwords(String content) {
        Map<String, Integer> map = new TreeMap<>();
        String[] words = content.split(" ");
        for (String word : words) {
            map.put(word, map.getOrDefault(word, 0) + 1);
        }
        return map;
    }

    //按照出现次数排序
    public String sortByValue(Map<String, Integer> map) {
        List<Map.Entry<String, Integer>> list = new ArrayList<>(map.entrySet());
        list.sort((o1, o2) -> o1.getValue().compareTo(o2.getValue()));
        StringBuilder sb = new StringBuilder();
        for (Map.Entry<String, Integer> entry : list) {
            sb.append(entry.getKey() + " ");
        }
        return sb.toString().trim();
    }

    //将内容写入文件
    public void writeToText(String content, String filepath) {
        try {
            File file = new File(filepath);
            if (!file.exists()) {
                file.createNewFile();
            }
            FileWriter fw = new FileWriter(file);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write(content);
            bw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

        }
        return content;
    }finally {
        close(buf);
    }
}

/**关闭输入流**/
private void close(Closeable inout) {
    if (inout != null) {
        try {
            inout.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

/**将文段用空格分隔**/
private String[] splitContentToWords(String content) {
    String contents[] = content.split(" ");
    return contents;
}

/**传入的String【】转换成TreeMap<words,times>**/
public TreeMap<String, Integer> countwords(String content) {
    TreeMap<String,Integer> map = new TreeMap<String, Integer>();
    Integer temp = null;
    String contents[] = splitContentToWords(content);
    for(int i = 0;i < contents.length;i++) {
        temp = map.put(contents[i], 1);
        if(temp !=null) {//表示同key值已经存在,temp会返回values值，加一表示又出现一次
            temp = temp+1;
            map.put(contents[i], temp);
        }
    }
    //map中有可能包含"", 所以移除掉
    map.remove("");
    return map;
}

/**将treemap重新排序 按照value从大到小输出**/
public String sortByValue(TreeMap<String, Integer> treeMap){
    String outstr = new String();
    List<Entry<String,Integer>> arrayList = new
ArrayList<Entry<String,Integer>>(treeMap.entrySet());
    //给sort传入一个Comparator实例，实现自定义排序，Comparator只是一个挽救的类(接口)
    MyComparator myComparator =new MyComparator();
    Collections.sort(arrayList,myComparator);
    for(Entry<String,Integer> mapper: arrayList){
        outstr = outstr + mapper.getKey() + "    "+mapper.getValue()+ "\r\n";
    }
    return outstr;
}

```

```

    /**将处理得到的字符串写入文件***/
    public void writeToText(String content,String filepath) throws IOException
    {
        File file = new File(filepath);
        OutputStream output = new FileOutputStream(file);
        //将OutputStream类对象传递给OutputStreamWriter类的构造方法，而后向上转型为Writer
        Writer wrt = new OutputStreamWriter(output);
        wrt.write(content);
        wrt.close();
    }
}

```

```

package container_IO;

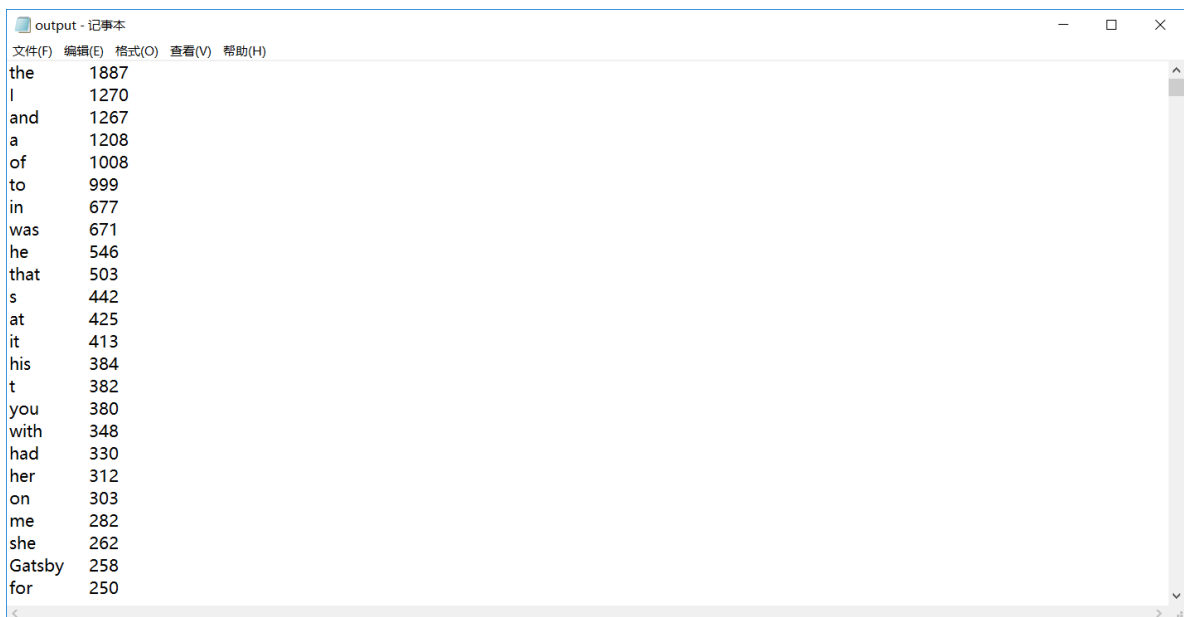
import java.util.Comparator;
import java.util.Map;
import java.util.Map.Entry;

//一个类继承了Comparator接口，方便后期更改需求
public class MyComparator implements Comparator<Map.Entry<String,Integer>>{

    @Override
    public int compare(Entry<String, Integer> arg0, Entry<String, Integer> arg1)
    {
        // TODO Auto-generated method stub
        return arg1.getValue()-arg0.getValue();
    }
}

```

实验实现



```

output - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
the      1887
I        1270
and      1267
a        1208
of       1008
to       999
in       677
was      671
he       546
that     503
s        442
at       425
it       413
his      384
t        382
you      380
with     348
had      330
her      312
on       303
me       282
she      262
Gatsby  258
for      250

```