**一、可行性分析**

1）操作可行性  
     操作可行性是指系统的操作方式在这个用户组织内行得通吗？本次开发只是简单的一个词频统计，不需要太大的人力或是资源。  
2） 经济可行性  
      经济可行性是从经济的角度分析系统的规划方案有无实现的可能性和开发的价值。本系统是自主开发的，用于自己的作业设计，所以不考虑资金上的流动，故而不存在经济不可行得问题。  
3） 技术可行性  
      技术上的可行性要考虑将来采用的硬件和软件技术能否满足用户提出的要求。目前，使用eclipse软件完全可以，所以，也不存在技术可行性的问题。

**二、需求分析**

根据实验二可知，使用JAVA编程语言，独立完成一个英文文本词频统计的软件开发。其基本需求如下：

* 程序可读入任意英文文本文件，该文件中英文词数大于等于1个。
* 程序需要很壮健，能读取容纳英文原版《哈利波特》10万词以上的文章。
* 指定单词词频统计功能：用户可输入从该文本中想要查找词频的一个或任意多个英文单词，运行程序的统计功能可显示对应单词在文本中出现的次数和柱状图。
* 高频词统计功能：用户从键盘输入高频词输出的个数k，运行程序统计功能，可按文本中词频数降序显示前k个单词的词频及单词。
* 统计该文本所有单词数量及词频数，并能将单词及词频数按字典顺序输出到文件result.txt。

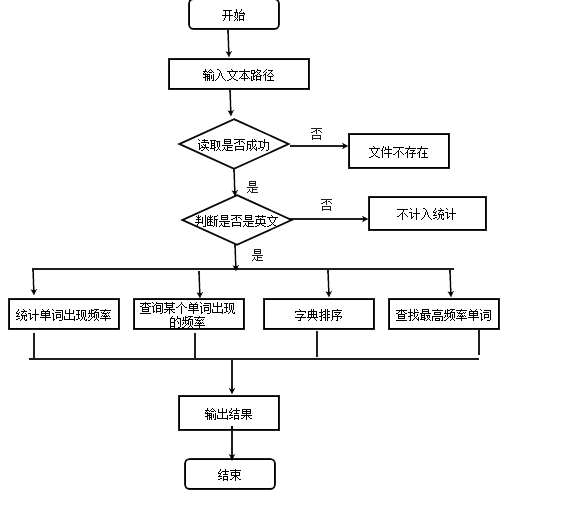
**三、功能实现**

   1)现统计词汇数目  
   2)查找指定词汇  
   3)查找高频单词  
   4)实现字典排序

**四、开发环境**

|  |  |
| --- | --- |
| **操作系统** | Widnows 7 |
| **开发工具** | Eclipse |
| **JDK** | 1.8 |
| **流程图设计** | 迅捷流程图制作软件 |

**五、流程图**



**六、部分代码**

    该程序共有三个类  
1）包含主函数main的主类:Main.java  
2）单词词频统计和高频词统计的功能类: Wordcount.java，对用户输入的单词进行处理，查询单词出现的次数。  
3）排序功能类:WordEntity.java，对整个文本文件进行降序或者排序。  
                       **词频统计**

try {

BufferedReader reader = new BufferedReader(new FileReader(fileName));

String line = null;

TreeMap<String,Integer> tm = new TreeMap<String,Integer>();

while((line=reader.readLine())!=null){

line = line.toLowerCase();

String str[] = line.split("\\s+");

for(int i = 0; i<str.length; i++){

String word = str[i].trim();

if(tm.containsKey(word)){

tm.put(word, tm.get(word)+1);

}else{

tm.put(word, 1);

}

}

}

//输出我们想要的字符串格式

System.out.println("按字典序输出为：");

Iterator<Entry<String, Integer>> iterator=tm.entrySet().iterator();

while(iterator.hasNext())

{

System.out.println(iterator.next());

}

} catch (FileNotFoundException e) {

e.printStackTrace();

}catch (IOException e) {

e.printStackTrace();

}

**高频统计**

try {

BufferedReader br = new BufferedReader(new FileReader(fileName));

String s;

StringBuffer sb = new StringBuffer();

while ((s = br.readLine()) != null) {

sb.append(s);

}

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

StringTokenizer st = new StringTokenizer(sb.toString(),",.! \n");

while (st.hasMoreTokens()) {

String letter = st.nextToken().trim();

int count;

if (!map.containsKey(letter)) {

count = 1;

} else {

count = map.get(letter).intValue() + 1;

}

map.put(letter,count);

}

Set<WordEntity> set = new TreeSet<WordEntity>();

for (String key : map.keySet()) {

set.add(new WordEntity(key,map.get(key)));

}

System.out.println("本文中出现频率最高的单词为：");

for(int i=0;i<rulerStep;i++){

g2.drawString((400-20\*i)+"个", 8, topMargin+rulerStep\*i);//绘制Y轴上的数据

}

g2.setColor(Color.PINK);

int m=0;

for(int i = 0;i<st.length;i++){

int value = tm.get(st[i]);

int step = (m+1)\*40;//设置每隔柱形图的水平间隔为40

g2.fillRoundRect(leftMargin+step\*2,Height-value, 40, value, 40, 10);//绘制每个柱状条

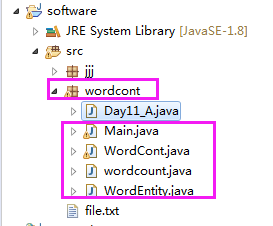
g2.drawString(st[i], leftMargin+step\*2, Height-value-5); //标识每个柱状条

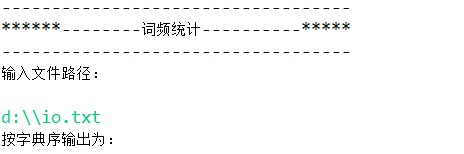
m++;

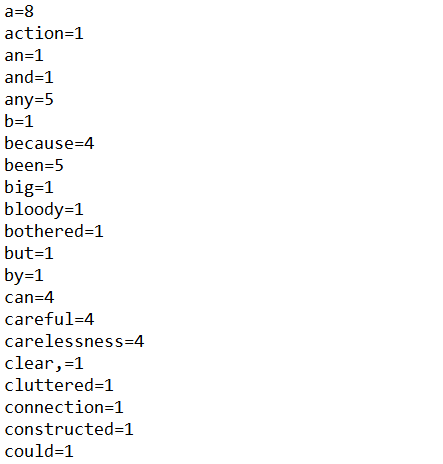
}\*/

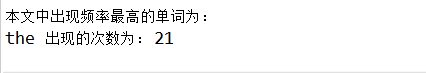
但是由于本人能力有限，之前对图形化的设计方面不太了解，最后结果没能呈现出来。

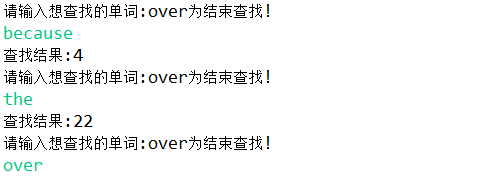
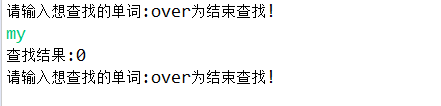
**七、测试运行**



**开始运行**  


**统计频率，并按字典排序输出**  


**统计出现频率次数最高**  


**查找单词功能**  
  


**八、PSP**

|  |  |  |  |
| --- | --- | --- | --- |
| PSP2.1 | 任务内容 | 计划共完成需要的时间（min） | 实际完成需要的时间（min） |
| Planning | 计划 | 15 | 10 |
| Estimate | 估计这个任务需要的时间，并规划大致工作步骤 | 15 | 10 |
| Development | 开发 | 180 | 220 |
| Analysis | 需求分析（包括学习新技术） | 6 | 8 |
| Design Spec | 生成设计文档 | 30 | 20 |
| Design Review | 设计复审 | 15 | 10 |
| Coding Standard | 代码规范（为目前的开发制作合适的规范） | 3 | 3 |
| Design | 具体设计 | 10 | 10 |
| Coding | 具体编码 | 50 | 60 |
| Code Review | 代码复审 | 5 | 8 |
| Test | 测试（自我测试、修改代码，提交修改） | 8 | 6 |
| Reporting | 报告 | 9 | 8 |
| Test Report | 测试报告 | 1 | 2 |
| Size Measurement | 事后总结，并提出过程改进计划 | 5 | 5 |

**九、总结**

|  |
| --- |
| 自己虽然之前编写过程序，也编写过一些小项目，但是，因为时间久了不动手导致在此次编程中 出现了很多问题，很多方面都出现了许许多多的问题。导致自己又回过头重新去学习，花费了大量 的时间。但是，由于自己的编程能力有限，柱形图的功能未能实现。 |