

# JAVA 编程进阶上机报告



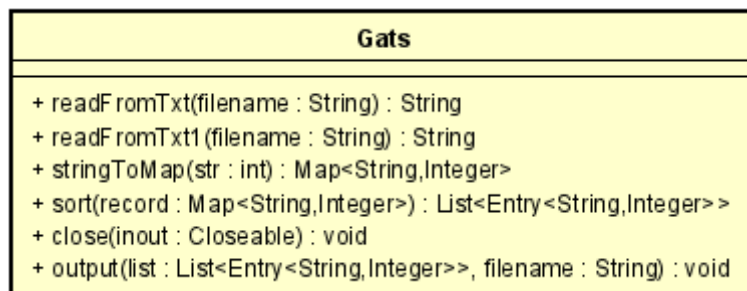
学 院 \_\_\_\_\_ 智能与计算学部 \_\_\_\_\_  
专 业 \_\_\_\_\_ 软件工程 \_\_\_\_\_  
班 级 \_\_\_\_\_ 2 班 \_\_\_\_\_  
学 号 \_\_\_\_\_ 3018216095 \_\_\_\_\_  
姓 名 \_\_\_\_\_ 郭紫珊 \_\_\_\_\_

# Lab 2 Container and IO

实验要求将文本中的单词按照出现的频率倒序排列。我的思路如下：

- (1) 把txt文件读取到一个String中
- (2) 将String分割成一个个单词，储存在数组中
- (3) 遍历数组，使用hashMap<String, Integer>，如果有这个单词就在原来的值上加一，如果没有，就新加入这个单词，把值设置为1。
- (4) 创建一个List<Entry<String,Integer>>，实现Comparable的接口，调用sort对频率排序。
- (5) 依次输出单词和他的频率到文件中。

然后因为整个实验只有一个类，所以只画了类图：



首先将txt文件读取到String中。有很多种方法，我想出来了两种，如下

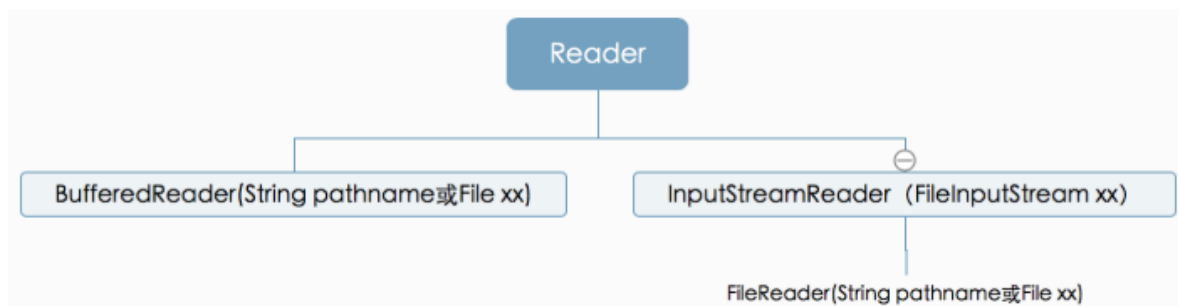
```
private static String readFromTxt(String filename) throws Exception {
    Reader reader = null;
    try {
        StringBuffer buf = new StringBuffer();
        char[] chars = new char[1024];
        reader = new FileReader(filename);
        int readed = reader.read(chars);
        while (readed != -1) {
            buf.append(chars, 0, readed);
            readed = reader.read(chars);
        }
        return buf.toString();
    } finally {
        close(reader);
    }
}
```

第一种是直接使用reader类，调用read ( ) 方法。

```
private static String readFromTxt1(String filename) throws IOException {
    StringBuffer buffer = new StringBuffer();
    BufferedReader bf= new BufferedReader(new FileReader(filename));
    String s = null;
    while((s = bf.readLine())!=null){//使用readLine方法，一次读一行
        buffer.append(s.trim());
    }
    return buffer.toString();
}
```

第二种是使用FileReader，然后转成BufferedReader，调用readLine（）方法。

这两个方法涉及很多类，比如同属于读取字节流的BufferedReader/FileReader等，他们的关系可以由下图表示：



所以通过BufferedReader/FileReader/InputStreamReader这三个方法读出来的文件，都可以直接输出字符。但是他们有很多差别。

InputStreamReader：可以指定字符编码格式,入参传递InputStream对象：

```
InputStreamReader inputstreamreader1=new InputStreamReader(System.in);
InputStreamReader inputstreamreader2=new InputStreamReader(new
FileInputStream("/opt/xxx"));
InputStreamReader inputstreamreader3=new InputStreamReader(new
FileInputStream(new File("/opt/xxx")));
```

FileReader：入参直接传递文件pathname或File对象

```
FileReader f1=new FileReader("/opt/xxx.txt");
FileReader f2=new FileReader(new File("/opt/xxx.txt"));
```

BufferedReader：入参有Reader对象和缓冲区大小（可不写）

从缓存区中读取字符流，提高效率；缓冲区大小：默认8192，默认不需要传递。最好都使用这个类去读取文件：

```
BufferedReader buffered_filereader=new BufferedReader(new FileReader(filename));
BufferedReader buffered_inputstreamreader=new BufferedReader(new
InputStreamReader(fileinputstream));
```

另外，BufferedReader对象使用readLine()方法判断字符串是否为null判断是否为文件末尾，Reader子类InputStreamReader和FileReader使用read()方法判断是否为-1，来判断是否为文件末尾。

所以从上面来看，如果是读取file类的文件，InputStreamReader，FileReader都可以，然后再转成BufferedReader对象，使用StringBuffer类将每次读取到的内容拼接在一起就可以了。

接下来就是把这个String拆成一个一个的单词，了不起的盖茨比这个txt里面没有逗号句号等标点符号，分割单词使用的是空格。但是注意到有的单词之间不止一个空格，所以使用split（）函数的时候正则表达式应当填写\s+(一个或多个空格)。遍历单词数组，使用map把单词和他对应的频率储存起来。

```
private static Map<String,Integer> stringToMap(String str){
    Map<String,Integer> map = new HashMap<>();
    String contents[] = str.split("\\s+");
    for(int i=0;i<contents.length;i++) {
        String temp = contents[i];
        //System.out.print(i+"----"+contents[i]);
        if(map.containsKey(temp)) {
            Integer ex = map.get(temp)+1;
            map.put(temp, ex);
        }else {
            map.put(temp,1);
        }
    }
    return map;
}
```

查阅资料，这里不仅仅可以用split（）函数来拆分字符串，还可以用StringTokenizer 类。**Java StringTokenizer** 属于 java.util 包，用于分隔字符串。StringTokenizer(String str, String delim)构造一个用来解析 str 的 StringTokenizer 对象，并提供一个指定的分隔符。

```
private static Map<String,Integer> stringToMap(String str){
    Map<String,Integer> map = new HashMap<>();
    StringTokenizer st = new StringTokenizer(str,"\\s+");
    while(st.hasMoreElements()) {
        String str = st.nextToken().toLowerCase();
        if(map.containsKey(str)) {
            Integer ex = map.get(str)+1;
            map.put(str, ex);
        }else {
            map.put(str,1);
        }
    }
    return map;
}
```

将单词存到map里面之后，就要对他们进行频率排序。但是存在map中的元素，按照普通的方法都是不容易排序的，可能排好了数字的顺序，但是却很难找出对应的单词顺序。在Map类设计是，提供了一个嵌套接口（static修饰的接口）：Entry。Entry将键值对的对应关系封装成了对象，即键值对对象，这样我们在遍历Map集合时，就可以从每一个键值对（Entry）对象中获取对应的键与对应的值。然后实现Comparable接口,重写compareTo()方法，就可以利用map类提供的sort（）方法。

```

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>>() {
        public int compare(Entry<String, Integer> o1, Entry<String,
    Integer> o2) {
            return o2.getValue().compareTo(o1.getValue());
        }
    });
    return list;
}

```

排序完了之后结果会储存在一个以键值对为元素的list里面。

接下来就是遍历list，然后将结果写入output文件即可。需要注意的就是如果想要在文件里面写入回车，单个的\r,\n都是不行的，要写\r\n,顺序不能变。还要注意处理一下没有output文件的异常，先检测有没有output文件，没有就创建一个。

```

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()+"\r\n");
    }
    //将字符流先写成字节流再写入
    byte data[] = temp.toString().getBytes();
    os.write(data);
    os.close();
}

```

多个方法涉及到打开文件，所以不要忘记关闭文件。

```

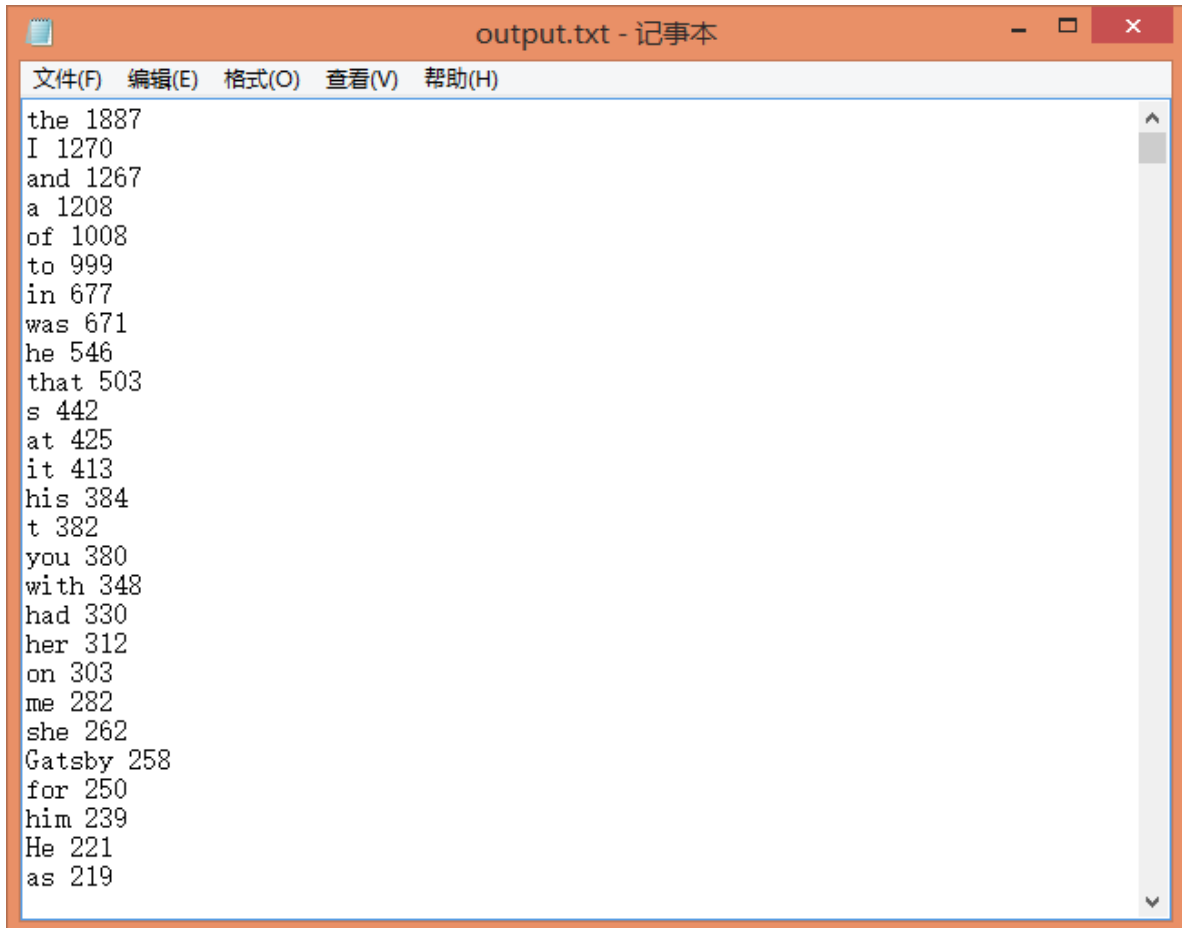
private static void close(Closeable inout) {
    if (inout != null) {
        try {
            inout.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

测试代码：

```
public static void main(String args[]) throws Exception {  
    Map<String, Integer> map = new HashMap<>();  
    String txt = "";  
    String input = "C:/Users/lenovo/Desktop/theGreatGatsby.txt";  
    String output = "C:/Users/lenovo/Desktop/output.txt";  
    txt = readFromTxt(input);  
    map = stringToMap(txt);  
    List<Entry<String,Integer>> list = sort(map);  
    output(list,output);  
}
```

运行结果：



```
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  
him 239  
He 221  
as 219
```

源代码如下：

```
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.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
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;

public class Gats {
    private static String readFromTxt(String filename) throws Exception {
        Reader reader = null;
        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);
            }
            return buf.toString();
        } finally {
            close(reader);
        }
    }

    private static Map<String,Integer> stringToMap(String str){
        Map<String,Integer> map = new HashMap<>();
        String contents[] = str.split("\\s+");
        for(int i=0;i<contents.length;i++) {
            String temp = contents[i];
            //System.out.print(i+"----"+contents[i]);
            if(map.containsKey(temp)) {
                Integer ex = map.get(temp)+1;
                map.put(temp, ex);
            }else {
                map.put(temp,1);
            }
        }
        return map;
    }

    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>>() {
            public int compare(Entry<String, Integer> o1, Entry<String,
            Integer> o2) {
                return o2.getValue().compareTo(o1.getValue());
            }
        });
        return list;
    }

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

```

```

    }
}

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()+"\r\n");
    }
    byte data[] = temp.toString().getBytes();
    os.write(data);
    os.close();
}

public static void main(String args[]) throws Exception {
    Map<String, Integer> map = new HashMap<>();
    String txt = "";
    String input = "C:/Users/lenovo/Desktop/theGreatGatsby.txt";
    String output = "C:/Users/lenovo/Desktop/output.txt";
    txt = readFromTxt(input);
    map = stringToMap(txt);
    List<Entry<String,Integer>> list = sort(map);
    output(list,output);
}
}

```