天津大学

JAVA 进阶编程第二次实验报告

学　　院　智能与计算学部

专　　业　软件工程

年　　级　17 级

姓　　名　王春杉

2019 年 3 月 21 日

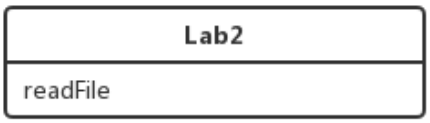
JAVA 进阶编程第二次实验报告

一、需求分析（描述具体需求）

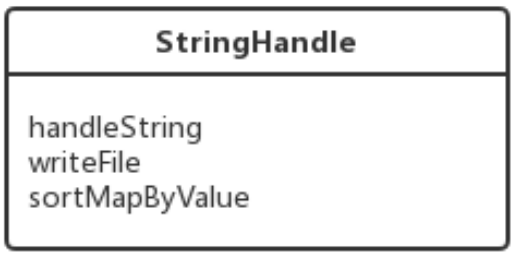
统计所给书目：了不起的盖茨比 中各个单词出现的频次
需要文件的输入输出，以及字符串处理
按照固定格式（单词 频次）写入到 output.txt 文件 （output 位于可执行文件同级目录：data 目录下）

二、概要设计（简单描述设计思路，配合 UML 图）

共两个类: Lab2 StringHandle
Lab2 为主类，用于读取文件和 main 函数， readFile 为读取文件的函数



StringHandle 为字符串处理类，包含函数：
handleString 处理字符串，
writeFile 写入文件
sortMapByValue 通过 Value 值降序排序 Map



三、详细设计（详细描述具体如何实现，附代码及说明）

具体实现：在 Lab2 类中利用 readFile 函数读取文本文件内容，并创建 StringHandle 类来处理文本内容，统计个数后保存在 HashMap 中，并利用 StringHandle 类将 Map 中的数据写入 output.txt 文件

Lab2 类：

readFile: 简单的文件读入

```
public static String readFile(String fileName){
    String content = "00";
    File file = new File(fileName);
    try {
        if (file.exists()){
            InputStream is = new FileInputStream(file);
            byte data[] = new byte[10000000];
            int foot = 0;
            int temp = 0;
            while((temp = is.read())!=-1) {
                data[foot++] = (byte) temp;
            }
            is.close();
            content = new String(data,0,foot);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return content;
}
```

StringHandle 类：

handleString: 将读取的文本内容处理后，存入 HashMap，再将 Map 进行排序，排序后将数据写入 output.tet 文件

字符处理：替换文本文件的标点为空格，保留文本文件的空格、英文字母，再按空格拆分初步处理后的文本

```java
public void handleString(String content) {
    Map<String, Integer> map = new HashMap<>();
    StringBuffer sBuffer = new StringBuffer();
    char[] temp = content.toCharArray();
    for(char i: temp) {
        if((i >= 33 && i <= 44) || (i >= 46 && i <= 47)) {
            sBuffer.append(" ");
        }
        if(i == ' ' || (i >= 'a' && i <= 'z') || (i >= 'A' && i <= 'Z')) {
            sBuffer.append(i);
        }
    }
    content = sBuffer.toString();
    content = content.toLowerCase();

    System.out.println(content);
    String[] contents = content.split(" ");
    for(int i = 0; i<contents.length; i++) {
        if(map.containsKey(contents[i])) {
            int newVal = map.get(contents[i]) + 1;
            map.put(contents[i], newVal);
        }else {
            map.put(contents[i], 1);
        }
    }
    map = StringHandle.sortMapByValue(map);
    String outputFileName = StringHandle.class.getResource("").getPath();
    try {
        outputFileName = URLDecoder.decode(outputFileName, "UTF-8"); // 将中文路径转为
        outputFileName = outputFileName + "data/output.txt";
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    for(Map.Entry<String, Integer> entry : map.entrySet()) {
        String word = entry.getKey();
        int num = entry.getValue();
        String toWrite = word + " " + num + "\n";
        try {
            StringHandle.writeFile(outputFileName, toWrite);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

writeFile： 将数据写入文件

```java
public static void writeFile(String fileName, String content) throws Exception {
    File outFile = new File(fileName);
    if(!outFile.getParentFile().exists()) {
        outFile.getParentFile().mkdirs();
    }
    FileOutputStream write = new FileOutputStream(outFile, true);
    byte[] data = content.getBytes();

    write.write(data);
    write.close();
}
```

sortMapByValue: 通过 Value 值将 map 降序排序

利用排序类 MapValueComparator，利用 ArrayList 的 Collections.sort 函数排序

```
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;
    int flag = 0;
    while (iter.hasNext()) {
        if(flag == 0) {
            flag = 1;
            tmpEntry = iter.next();
        }else {
            tmpEntry = iter.next();
            sortedMap.put(tmpEntry.getKey(), tmpEntry.getValue());
        }

    }
    return sortedMap;
}
```

四、调试分析（在实验过程中遇到的问题以及如何解决）

在编写代码时，遇到相对路径读取的相关问题...

1. 中文路径乱码问题：通过转义成 UTF-8 解决
2. 路径读取错误问题：修改了很多方法，最终确定了现有方法

传入文本时乱码问题：

暴力保留固定文本

五、测试结果（描述输入和输出）

输入：

txt 文本

列出部分输出结果：(output.txt 文件)

```
the 1996
and 1343
i 1289
a 1272
of 1025
to 1007
he 781
in 710
was 677
it 563
that 506
you 494
at 459
```

六、总结

本实验主要是文本文件的处理，利用 String 内部函数可以方便处理
通过该实验，熟悉了文件的输入输出，以及研究了相对路径
也使用了简单的异常处理