# FindIt(帮我筛简历)实验报告

## Table of Contents

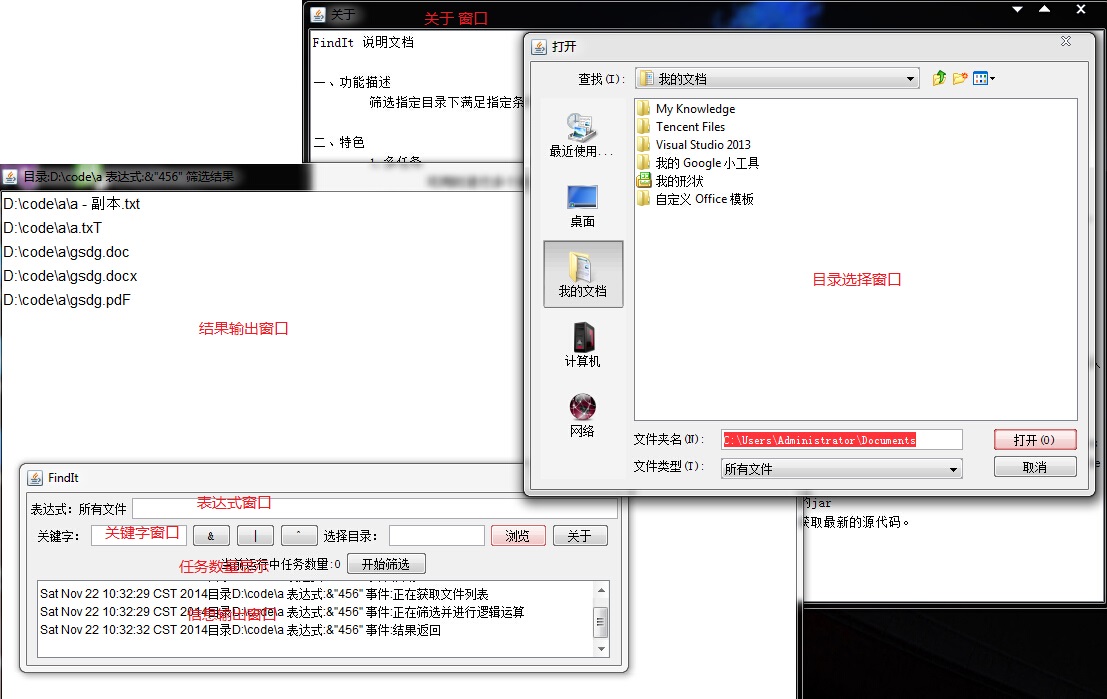
* 1 需求分析
* 2 整体设计
* 3 实现
* 4 实验测试
* 5 关于本程序
* 6 组员分工
* 7 项目地址

## 1 需求分析

* 题目 人工筛取海量简历费时费力，用Java实现一个简历自动筛选软件，实现以下功能：
  + 需要有界面
  + 能够设置一个目录，软件对该目录下的所有简历进行筛选
  + 能够对word和pdf格式的简历进行筛选
  + 筛选时，从软件界面指定关键字，然后读取所有word文档和pdf文档的内容，查找其中是否存在该关键字，如果存在，则将文件路径名显示在界面上。筛选结果是一系列文件路径名组成的列表。（对word和pdf文档内容的读取，可以借助第三方的程序库实现）
  + 支持同时指定多个关键字（至少3个），并同时依据这3个关键字进行筛选。
  + 不仅支持英文关键字，还要支持中文关键字。
  + 可选功能：支持对关键字的排除，例如，能够通过“-西安电子科技大学”筛 选出所有不包含“西安电子科技大学”关键字的简历.

## 2 整体设计

* 各部分设计
  + GUI



* + 总体设计



* 难点及解决方案
  + 读取PDF及Doc（Docx）格式的文档
    - 通过互联网搜索，引入外部包（POI，PDFBOX），已解决。
  + 逻辑查找
    - 设计一个语法分析器和一个结果组合器，解析并执行表达式中的逻辑符 号
  + 表达式中是否可以出现括号
    - 这个问题困扰了我好久，后来决定使用正则匹配来替代之。（使用正则 匹配后理论来说连关键字都可以省略，但是考虑到并非所有用户都可以 写出正确的正则表达式，故保留了关键字的与或非运算）

## 3 实现

1. 设计时采用自上而下的设计方案，先从总体把握设计，然后设计细节，但在 实现过程用，我使用自下而上的实现方式，每实现一部分都进行测试，降低 后期维护成本。以下为测试代码：
2. 1: package findIt;
3. 2:
4. 3: import java.lang.reflect.InvocationTargetException;
5. 4: import javax.swing.SwingUtilities;
6. 5: import javax.swing.UIManager;
7. 6: import javax.swing.UnsupportedLookAndFeelException;
8. 7:
9. 8:
10. 9: /\*\*
11. 10: \* 测试grammerAnalysis.keyWordExtraction()
12. 11: \* @author SkyFire
13. 12: \*
14. 13: \*/
15. 14: */\**
16. 15: *public class test222{*
17. 16:  *public static void main(String [] args){*
18. 17:  *ArrayList<String> arrl=new ArrayList<String>();*
19. 18:  *Scanner in=new Scanner(System.in);*
20. 19:  *String tempStr=in.next();*
21. 20:  *in.close();*
22. 21:  *tempStr=grammerAnalysis.keyWordExtraction(tempStr,arrl);*
23. 22:  *for(String st:arrl){*
24. 23:  *System.out.println(st);*
25. 24:  *}*
26. 25:  *System.out.println(tempStr);*
27. 26:  *}*
28. 27: *}*
29. 28: *\*/*
30. 29:
31. 30:
32. 31: /\*\*
33. 32: \* 测试FindItFileList.getFileList()
34. 33: \* @author SkyFire
35. 34: \*
36. 35: \*/
37. 36: */\**
38. 37: *public class test222{*
39. 38:  *public static void main(String [] args){*
40. 39:  *ArrayList<String> arrl=new ArrayList<String>();*
41. 40:  *Scanner in=new Scanner(System.in);*
42. 41:  *String tempStr=in.next();*
43. 42:  *in.close();*
44. 43:  *arrl=FindItFileList.getFileList(tempStr);*
45. 44:  *for(String st:arrl){*
46. 45:  *System.out.println(st);*
47. 46:  *}*
48. 47:  *}*
49. 48: *}*
50. 49: *\*/*
51. 50:
52. 51: /\*\*
53. 52: \* 读取pdf测试
54. 53: \* @author SkyFire
55. 54: \*
56. 55: \*/
57. 56:
58. 57: */\**
59. 58: *public class test222{*
60. 59:  *public static void main(String [] args){*
61. 60:  *Scanner in=new Scanner(System.in);*
62. 61:  *String tempStr=in.next();*
63. 62:  *in.close();*
64. 63:  *String arrl=readFile.readPDF(tempStr);*
65. 64:  *System.out.println(arrl);*
66. 65:  *}*
67. 66: *}*
68. 67:
69. 68: *\*/*
70. 69:
71. 70: /\*\*
72. 71: \* 读word
73. 72: \* @author SkyFire
74. 73: \*
75. 74: \*/
76. 75: */\**
77. 76:
78. 77: *public class test222{*
79. 78:  *public static void main(String [] args){*
80. 79:  *Scanner in=new Scanner(System.in);*
81. 80:  *String tempStr=in.next();*
82. 81:  *in.close();*
83. 82:  *String arrl=null;*
84. 83:  *try{*
85. 84:  *arrl=readFile.readOffice(tempStr);*
86. 85:  *}catch(XmlException e){*
87. 86:  *}catch(OpenXML4JException e){*
88. 87:  *}catch(FileNotFoundException e){*
89. 88:  *}catch(Exception e){*
90. 89:  *}*
91. 90:  *System.out.println(arrl);*
92. 91:  *}*
93. 92: *}*
94. 93: *\*/*
95. 94:
96. 95: /\*\*
97. 96: \* 测试readText
98. 97: \* @author SkyFire
99. 98: \*
100. 99: \*/
101. 100: */\**
102. 101: *public class test222{*
103. 102:  *public static void main(String [] args){*
104. 103:  *Scanner in=new Scanner(System.in);*
105. 104:  *String tempStr=in.next();*
106. 105:  *in.close();*
107. 106:  *String arrl=null;*
108. 107:  *try{*
109. 108:  *arrl=readFile.readText(tempStr);*
110. 109:  *}catch(FileNotFoundException e){*
111. 110:  *}catch(Exception e){*
112. 111:  *}*
113. 112:  *System.out.println(arrl);*
114. 113:  *}*
115. 114: *}*
116. 115: *\*/*
117. 116:
118. 117: /\*\*
119. 118: \* 主类
120. 119: \* @author SkyFire
121. 120: \* @version 1.0
122. 121: \*/
123. 122: public class FindItMain{
124. 123: public static void main(String [] args) throws InvocationTargetException, InterruptedException{
125. 124: SwingUtilities.invokeAndWait(new Runnable() {
126. 125: @Override
127. 126: public void run() {
128. 127: try {
129. 128: UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
130. 129: } catch (ClassNotFoundException e) {
131. 130: e.printStackTrace();
132. 131: } catch (InstantiationException e) {
133. 132: e.printStackTrace();
134. 133: } catch (IllegalAccessException e) {
135. 134: e.printStackTrace();
136. 135: } catch (UnsupportedLookAndFeelException e) {
137. 136: e.printStackTrace();
138. 137: }
139. 138: }
140. 139: });
141. 140: FindItGui fgui=new FindItGui();
142. 141: }
143. 142: }

后来将此代码直接用于完成版本

1. 关键代码注释
   1. 语法分析器
   2. 1: package findIt;
   3. 2: import java.io.File;
   4. 3: import java.io.IOException;
   5. 4: import java.util.ArrayList;
   6. 5:
   7. 6: import org.apache.poi.openxml4j.exceptions.OpenXML4JException;
   8. 7: import org.apache.xmlbeans.XmlException;
   9. 8:
   10. 9: /\*\*
   11. 10: \* 语法分析器
   12. 11: \* @author SkyFire
   13. 12: \*/
   14. 13:
   15. 14: class FindItAnalysis{
   16. 15: private String expression;
   17. 16: private String dir;
   18. 17: private String op;
   19. 18: private ArrayList<String> keyWordVec=new ArrayList<String>();
   20. 19: private ArrayList<String> fileList;
   21. 20: private ArrayList<String> result;
   22. 21: /\*\*
   23. 22: \* @param gui GUI实例
   24. 23: \*/
   25. 24: void startAnalysis(FindItGui gui){
   26. 25: */\**
   27. 26:  *\* 获取逻辑表达式褐和目录*
   28. 27:  *\*/*
   29. 28: expression=new String(gui.getExpression());
   30. 29: dir=new String(gui.getDir());
   31. 30: */\**
   32. 31:  *\* 测试目录是否存在*
   33. 32:  *\*/*
   34. 33: File file =new File(dir);
   35. 34: if (!(file.exists()&&file.isDirectory())){
   36. 35: gui.errorOut(expression,dir,"目录不存在！",1);
   37. 36: return;
   38. 37: }
   39. 38: */\**
   40. 39:  *\* 检验表达式中的引号*
   41. 40:  *\*/*
   42. 41: if(grammerAnalysis.quotesNum(expression)%2==1){
   43. 42: gui.errorOut(expression,dir,"逻辑表达式中引号数量有误！",1);
   44. 43: return;
   45. 44: }
   46. 45: */\**
   47. 46:  *\* 提取关键字和逻辑操作符*
   48. 47:  *\*/*
   49. 48: op=grammerAnalysis.keyWordExtraction(expression,keyWordVec);
   50. 49: */\**
   51. 50:  *\* 测试逻辑操作符数量*
   52. 51:  *\*/*
   53. 52: if(op.length()!=keyWordVec.size()){
   54. 53: gui.errorOut(expression,dir,"逻辑表达式中操作符数量有误！",1);
   55. 54: return;
   56. 55: }
   57. 56: */\**
   58. 57:  *\* 测试逻辑操作符合法性*
   59. 58:  *\*/*
   60. 59: if(!grammerAnalysis.opTest(op)){
   61. 60: gui.errorOut(expression,dir,"逻辑表达式中操作符有误！",1);
   62. 61: return;
   63. 62: }
   64. 63: */\**
   65. 64:  *\* 获取文件列表*
   66. 65:  *\*/*
   67. 66: gui.receMsg(expression, dir, "正在获取文件列表");
   68. 67: fileList=FindItFileList.getFileList(dir);
   69. 68: */\**
   70. 69:  *\* 初始结果为所有文件*
   71. 70:  *\*/*
   72. 71: result=new ArrayList<String>(fileList);
   73. 72: */\**
   74. 73:  *\* 结果生成*
   75. 74:  *\*/*
   76. 75: gui.receMsg(expression, dir, "正在筛选并进行逻辑运算");
   77. 76: for(int i=0;i<op.length();++i){
   78. 77: try{
   79. 78: */\**
   80. 79:  *\* 获取并组合匹配结果*
   81. 80:  *\*/*
   82. 81: result=mergeIt.merge(result, FindItMatch.getMatch(fileList,(String) keyWordVec.toArray()[i]),op.charAt(i));
   83. 82: }catch(IOException e){
   84. 83: gui.errorOut(expression,dir,e.toString(),0);
   85. 84: }catch(OpenXML4JException e){
   86. 85: gui.errorOut(expression,dir,e.toString(),0);
   87. 86: }catch(XmlException e){
   88. 87: gui.errorOut(expression,dir,e.toString(),0);
   89. 88: }
   90. 89: }
   91. 90: */\**
   92. 91:  *\* 输出结果*
   93. 92:  *\*/*
   94. 93: gui.resultOut(expression,dir,result);
   95. 94: }
   96. 95: }
   97. 目录获取器
   98. 1: package findIt;
   99. 2: import java.io.File;
   100. 3: import java.util.ArrayList;
   101. 4: /\*\*
   102. 5: \* 获取目录列表
   103. 6: \* @author SkyFire
   104. 7: \*
   105. 8: \*/
   106. 9:
   107. 10: class FindItFileList{
   108. 11: /\*\*
   109. 12: \* 获取目录列表
   110. 13: \* @param dir 目录
   111. 14: \* @return 所有文件列表
   112. 15: \*/
   113. 16: static ArrayList<String> getFileList(String dir){
   114. 17: ArrayList<String> fileList=new ArrayList<String>();
   115. 18: File file=new File(dir);
   116. 19: if (file.isDirectory()) {
   117. 20: File[] dirFile = file.listFiles();
   118. 21: for (File f : dirFile) {
   119. 22: */\**
   120. 23:  *\* 如果是目录，则递归获取，加入当前列表*
   121. 24:  *\*/*
   122. 25: if (f.isDirectory()) {
   123. 26: fileList.addAll(getFileList(f.getAbsolutePath()));
   124. 27: } else {
   125. 28: fileList.add(f.getAbsolutePath());
   126. 29: }
   127. 30: }
   128. 31: }
   129. 32: return fileList;
   130. 33: }
   131. 34: }
   132. 结果组合器
   133. 1: package findIt;
   134. 2: import java.util.ArrayList;
   135. 3:
   136. 4: /\*\*
   137. 5: \* 结果组合器
   138. 6: \* @author SkyFire
   139. 7: \*
   140. 8: \*/
   141. 9: class mergeIt{
   142. 10: /\*\*
   143. 11: \* 逻辑运算
   144. 12: \* @param ls1 列表1
   145. 13: \* @param ls2 列表2
   146. 14: \* @param op 运算符
   147. 15: \* @return 逻辑运算结果
   148. 16: \*/
   149. 17: static ArrayList<String> merge(ArrayList<String> ls1,ArrayList<String> ls2,char op){
   150. 18: ArrayList<String> tempArr=new ArrayList<String>();
   151. 19: switch (op){
   152. 20: case '&':
   153. 21: */\**
   154. 22:  *\* 查找相同的，加入*
   155. 23:  *\*/*
   156. 24: tempArr=new ArrayList<String>();
   157. 25: outer:
   158. 26: for(String t1:ls1){
   159. 27: for(String t2:ls2){
   160. 28: if(t1.equals(t2)){
   161. 29: tempArr.add(t1);
   162. 30: continue outer;
   163. 31: }
   164. 32: }
   165. 33: }
   166. 34: break;
   167. 35: case '|':
   168. 36: */\**
   169. 37:  *\* 加入第一个列表，然后查找第一个列表中没有的，加入*
   170. 38:  *\*/*
   171. 39: tempArr=new ArrayList<String>(ls1);
   172. 40: outer:
   173. 41: for(String t2:ls2){
   174. 42: for(String t1:ls1){
   175. 43: if(t1.equals(t2)){
   176. 44: continue outer;
   177. 45: }
   178. 46: }
   179. 47: tempArr.add(t2);
   180. 48: }
   181. 49: break;
   182. 50: case '^':
   183. 51: */\**
   184. 52:  *\* 查找第二个列表中没有的，加入*
   185. 53:  *\*/*
   186. 54: tempArr=new ArrayList<String>();
   187. 55: outer:
   188. 56: for(String t1:ls1){
   189. 57: for(String t2:ls2){
   190. 58: if(t1.equals(t2)){
   191. 59: continue outer;
   192. 60: }
   193. 61: }
   194. 62: tempArr.add(t1);
   195. 63: }
   196. 64: break;
   197. 65: }
   198. 66: return tempArr;
   199. 67: }
   200. 68: }

## 4 实验测试

* 安全及稳定性：在设计过程中，已经进行各功能模块的测试，完成后自己写了几个文件进行 测试，目前（提交前）没有安全及稳定性问题。
* 效率：在效率方面确实存在一些问题，筛选每个关键字都需要对所有文件进 行重新读入，使用第三方库读取PDF及Doc文件效率也略低。
  + 解决方案及其可行性 开发过程中有着尝试将读文件任务并发处理，但是那样会需要更加庞大的 内存空间，会影响到程序的稳定性，所以不可行，正在寻找一种折中的方 案。
* 程序的可移植性以及可扩充性
  + 可移植性可以先不谈，因为java本来就是一种可移植编程语言
  + 可扩充性 这个程序的功能扩充主要是对支持文件类型的种类进行扩充，增加文件类 型需要在readFile.java中添加读取某种类型文件的方法，以及在 findItMatch.java中添加对文件类型判断的代码

## 5 关于本程序

* + 1. 程序已经实现指定的基础功能及可选功能，并做了一些扩充

FindIt 说明文档

一、功能描述

筛选指定目录下满足指定条件的文档

二、特色

1.多任务

可同时进行多个筛选任务

2.逻辑筛选

可以将关键字通过‘&’、“|”、‘^’组合起来进行逻辑筛选，如查找同时含有“西安”、“大学”,但是不包括“外国语”的文档，则可以输入表达式：&"西安"&"大学"^"外国语"

3.支持多种文件格式筛选

目前支持.txt，.pdf，.doc，.docx，.ppt，.pptx，.xls，.xlsx文件格式的筛选

4.支持正则筛选

关键字可以为正则表达式，如：&"西安.\*大学" 可以筛选出含有“西安(任意字符)大学” 的所有结果

5.操作简单

对于一般关键字可以在关键字框直接输入关键字，点击任意逻辑符号按钮，将在表达式框里自动生成表达式，如：输入关键字“西安”，点击 & 按钮，会在表达式框内生成&"西安"

三、关于版权

此项目中引入的扩展JAR：pdfbox-app-1.8.7.jar、xmlbeans-2.3.0.jar、poi-3.11-beta3-20141111.jar、poi-examples-3.11-beta3-2.141111.jar、poi-excelant-3.11-beta3-20141111.jar、poi-ooxml-3.11-beta3-20141111.jar、poi-ooxml-schemas-3.11-beta3-20141111.jar、poi-scratchpad-3.11-beta3-20141111.jar。

以上包的版权属于原作者，在此向开发者们致敬，感谢你们开发出如此强大的jar

此项目为开源项目，可随时在<http://github.com/skyfireitdiy/FindIt>获取最新的源代码。

--SkyFire

## 6小组分工

## 1小组名称：Java研发小组-代号J

## 2小组成员：王茂斌（03121303），张世欣（03121311），杨林春（03121304），纪志鹏（03121305），王玉龙（03121307），刘易斯（03121039）

## 3工作模式：通过QQ讨论组方式讨论并分配任务

## 4分工：

## 王茂斌：整体设计，撰写文档，语法分析器编码，驱动编码，整体测试

## 张世欣：GUI设计，GUI编码

## 杨林春：文件匹配器编码（包括文件读取）

## 刘易斯：文件列表获取器编码

## 王玉龙：查找pdf，doc等格式读取方法，提供外部包及其使用方法

## 纪志鹏：各组件分析，测试

## 7 项目地址：<http://github.com/skyfireitdiy/FindIt>