Lab06

by Jason Ivey using: LaTeX & Dr Java.

November 2016

1 Goal

To read a textual file and declare which parenthesis are missing from the parenthesis pairs.

2 Lab06.bat

This lab looks better when run in CMD with color set to 02. So I wrote a .bat file that will run this program for you! It was an interesting experiment in another language.

Conditions to run the .bat or BATCH file:

- You must have the java PATH: C
- You must have my lab's name as Lab06.java
- You must find the directory in which my lab is stored

```
/**
 * This is a LAB assignment by Jason Ivey for Java II, a class by Prof. C. Servin.
 * The Author of this program would like to extend thanks to{
 * Valentine Bacerra, for his help in suggesting the use of the Java Stack<character>.
 * Terry Speciher for his suggestions in processing the text.
 * Professor Servin for his detailed explanations of stacks.
 * }
 * @author Jason Ivey
 */

import java.util.Scanner; //Imports for Scanner, Stack, and file operations.
import java.util.Stack;
import java.io.*;

/**
 * Main Class, makes use of FileNotFoundException, InvalidFormatException, Scanner, Stack and util
     .IO.
 */
public class Lab06{

  private static String error = ""; // String to hold error message
  private static final boolean DEBUG = false;
  /**
   * Main Method, accepting string arguments
   * @param args standard header String[]
   */
  public static void main(String args[]){
    if (DEBUG)
      System.out.println("Before input error = " + error);
    Scanner s = new Scanner(System.in); // interface with keyboard

    String fileName = "First Iteration"; // filename, initialized to "First Iteration" during
         first iteration.
    //What follows is a welcome screen!
    System.out.println(" _____                    _  _              _      _____\n" +
                       "|  __ \\                   | | | |            (_)     |  __ \\
                       \n"+
                       "| |__) |_ _ _ __ ___ _ __ | |_| |__   ___  ___ _ ___  | |__) |_ _ _ __ ___
                       ___ \n"+
                       "|  ___/ _` | '__/ _ \\ '_ \\| __| '_ \\ / _ \\/ __| / __| |  ___/ _` | '__
                       / __|/ _ \\ \n"+
```

```
37                          "| |  | (-| | | |  --/ | | | |-| | | |  --/\\-- \\ \\-- \\ | |  | (-| | |
                                \\-- \\  --/\n"+
38                          "|-|   \\-- ,-|-|  \\---|-| |-|\\--|-| |-|\\---||---/-|---/ |-|   \\-- ,-|-|
                                |---/\\---|\n"+
39
40                          ".                          A Lab Assignment by Jason Ivey
                                                .\n" +
41                          ".                With help from: Mr. Bacerra, Prof. Servin, Mr. Speicher,
                                    .\n"+
42                          ".                             Prof. Blando, and Oracle\u00AE!
                                                .\n"+
43                          ".          This LAB is optimized for Java, other source types may not work
                                    .    .\n" ); //Welcome message with ASCII art
44      do{ // main loop that will execute until sentinel value of an empty string is called.
45
46        try{ //try method for possible FileNotFoundException or InvalidFormatException.
47
48          System.out.print("\nPress \"[Enter]\"  to exit, type \"help\"  for help, or ENTER FILENAME
                : "); //Prompt for user input
49          fileName = s.nextLine();//Recieving user input through keyboard
50          if (DEBUG)
51            System.out.println("After input error = " + error + "\nfileName = " + fileName);
52          if (helpOrExit(fileName))
53          {
54            if (DEBUG)
55              System.out.println("HelpOrExit = true");
56  //Checking for sentinel value or help request, if help or exit it will recheck condition.
57            continue;
58          }
59
60          else if(fileName.contains(".")){//Checking input for file extension
61            if (DEBUG)
62              System.out.println("File name does contain \".\"");
63            Stack<Character> stack = read(fileName); //A new stack of the type character, utilizing
                  java's built in Stack
64            if (DEBUG)
65              System.out.println("Stack  after read = " + stack +"\nError = " + error);
66            parse(stack);// Parsing remnants of stack into error.
67            if (DEBUG)
68              System.out.println("Stack  after parse = " + stack +"\nError = " + error);
69            if(error.equals(""))//If no error then everything is okay!
70              System.out.println("Everything is Okay!");
71
72            else{ //else diagnos the file
73
74              System.out.println("----------------------Read Succesful----------------------\n"
75                                  +"Diagnosis: Missing these paranthesis [" +  error.substring(0,
                                        error.length()-1) + "]"); //length is minus one to remove
                                        extra comma
76            }
77          }
78
79          else{ //If extension is missing then file name is invalid
80            if(fileName.trim().length() >= 0)
81              throw new InvalidFormatException("\nOh no..."
82                                               +"\nFile Name is Invalid");
83          }
84        }
85        catch(InvalidFormatException e) //Bad format
86        {
87          System.out.println("\n"+e);
88          continue;
89        }
90        catch (FileNotFoundException e){ //Explains file is missing
91          System.out.println("\nOh no..."+"\nFile does not exist in the directory: " + System.
                getProperty("user.dir") +"");
92          continue;
93        }
94        error = "";
95      } while( fileName.trim().length() > 0); //Condition of main loop
96      s.close();
97    }
98    /**
99     * HelpOrExit Will check for help or exit request
100    * @param input String of input to be checked for KEY_WORD or exit request.
101    * @return Boolean Return true if an exit or a help was requested.
102    */
103   public static boolean helpOrExit(String input){
104     final String KEY_WORD = "help";//Keyword help to be checked for
105     if (input.trim().length() <= 0 ) //Exit request is empty String
106       return true;
```

```
107
108       else if (input.length() != 0 && input.substring( 0 , Math.min(input.length(), KEY_WORD.length
              () )
109                                                                 ).equalsIgnoreCase(KEY_WORD.substring( 0 , Math
                                                                        .min(input.length(), KEY_WORD.length() ) )
                                                                    ) )
110         /*Extremely complicated if statements explained by Frank Blando in Java I,
111          * Basically if someone were to only "hE" it would only compare the first
112          * two characters of the key word.
113          */
114       {
115         System.out.println("\n____This program allows you to check Java progams for missing
                parenthesis____"
116                            +"\n - Aplicaple parethesis include: \" (, ), <, >, {, }, [,\" and
                                   \"].\""
117                            +"\n - To exit pres \"[Enter]\""
118                            +"\n - Reading: \".ppt, .mov\" and other non text files may cause
                                   errors"
119                            +"\n - You may check any textual filetype, make sure and include the
                                   extension. "
120                            +"\n - \".java\" and \".txt\" are the best extensions to use."); //
                                   Helpful text
121         return true;
122       }
123       else return false; //If no request for help ro exit then false!
124     }
125     /**
126      * Read will read the specified file for characters and will remove characters with spouses
127      * @param fileName String of the name of the file to be read.
128      * @return Stack<charcater> Returns a stack of characters that did not get paired up with their
             spouses.
129      */
130     public static Stack<Character> read(String fileName) throws FileNotFoundException {
131       Stack<Character> temp = new Stack<Character>(); //temp Stack!
132       String buffer;//String to hold line value of file
133
134       File myFile = new File(fileName);//New file to be read!
135
136       if(!myFile.exists())//Wait... does it exist?
137         throw new FileNotFoundException();//NO! Then throw this!
138
139       Scanner fileReader = new Scanner(myFile);//Lets scan the contents of the file with this.
140
141       while(fileReader.hasNext()){//Loop to go through all input from file
142
143         buffer = fileReader.nextLine();//update buffer
144
145         for(int x = 0; x < buffer.length(); x++){ //For the length of the line
146           /*
147            * The problem of Boolean statements and loops was brought up in class and
148            * after trying to implement some pretty wacky answers I decided to call
149            * over Valentine and ask him how the heck I was supposed to fix this.
150            * He suggested I check the buffer then check the char at x for a comparitive
151            * after this just skip the loop and continue.
152            */
153           if(buffer.contains("if")          && (buffer.charAt(x) == '<' || buffer.charAt(x) == '>'))
154             continue;
155           else if(buffer.contains("return") && (buffer.charAt(x) == '<' || buffer.charAt(x) == '>'))
156             continue;
157           else if(buffer.contains("while")  && (buffer.charAt(x) == '<' || buffer.charAt(x) == '>'))
158             continue;
159           else if(buffer.contains("for")    && (buffer.charAt(x) == '<' || buffer.charAt(x) == '>'))
160             continue;
161
162           switch (buffer.charAt(x)){
163             /*This switch checks for parenthesis and pushes open parenthesis and pops
164              * open parenthesis when it encounters their spouses.
165              * If no spouse is found then error will have the opposite of the closing added to it
166              * (Because the spouse is the one who is missing).
167              */
168
169             case '(' : temp.push('(');
170             break;
171
172             case ')' :
173               if(!temp.empty() && temp.peek() == '(')
174               temp.pop();
175               else if(temp.empty())
176                 error = " (," + error;
177               break;
178
```

```
179             case '{' : temp.push('{');
180             break;
181
182             case '}' :
183               if(!temp.empty() &&temp.peek() == '{')
184               temp.pop();
185               else if(temp.empty())
186                 error = " {," + error;
187               break;
188
189             case '[' : temp.push('[');
190             break;
191
192             case ']' :
193               if(!temp.empty() && temp.peek() == '[')
194               temp.pop();
195               else if(temp.empty()|| temp.peek() != '<')
196                 error = " [," + error;
197               break;
198
199             case '<' : temp.push('<');
200             break;
201
202             case '>' :
203               if(!temp.empty() && temp.peek() == '<')
204               temp.pop();
205               else if(temp.empty() || temp.peek() != '<')
206
207
208                 error = " <," + error;
209               break;
210               //No default needed!!!
211           }
212         }
213       }
214       fileReader.close();
215       return temp;//lets give the caller back the temp they deserve!
216     }
217     /**
218      * parse Will read the stack for characters and assign the spouse(opposite) of the characters
            into error.
219      * @param s Stack<character> to be read.
220      *
221      */
222     public static void parse(Stack<Character> s){//This does the rest fo the parsing needed
223
224       while(!s.empty())//Do this till our stack is empty
225       {
226         /*
227          * Will basically set the opposite to error.
228          */
229         switch (s.pop()){//Switching on the character we just pooped
230           case '(' : error = " )," + error;
231           break;
232           case ')' : error = " (," + error;
233           break;
234           case '{' : error = " }," + error;
235           break;
236           case '}' : error = " {," + error;//Does the operation in reverse order because of the FILO
                nature of stacks
237           break;
238           case '[' : error = " ]," + error;
239           break;
240           case ']' : error = " [," + error;
241           break;
242           case '<' : error = " >," + error;
243           break;
244           case '>' : error = " <," + error;
245           break;
246           //No default needed
247         }
248       }
249     }
250   }
251   class InvalidFormatException extends Exception{//Nested class for exceptions
252     /**
253      * Constructor of Exception.
254      * @param s String to be printed as a message.
255      */
256     public  InvalidFormatException(String s){
257       super(s);
```

```
258     }
259 }
```

3 Here is the Lab06.bat source code!

```
1
2
3   @echo off
4   rem This is a comment!
5   rem Author: Jason Ivey,.bat file to run in command line.
6   mode con: cols=1080 lines=720
7   title ParenthesisParse
8
9   color 02
10
11  echo[
12
13  rem echo[ prints a blank line
14
15
16  echo  Where is Jason Ivey's Lab Folder? Find it in file explorer then copy the path here
17
18  SET /P X=Type the directory:
19
20  rem Set gets input
21
22
23  cd %X%
24
25  rem changing directory
26
27  :begin
28
29
30  rem Label for spaghetti code
31
32  for /l %%x in (1, 1, 4) do (echo[)
33
34
35  javac Lab06.bat
36
37  rem compiles Lab06.bat!
38
39
40  java Lab06
41
42  rem Runs Lab06
43
44
45  echo[
46
47  echo You have just run my lab! Awesome, would you like to run it again?
48
49  echo[
50
51  SET /P X=Type 'y' or 'n' or "starwars":
52
53  echo[
54
55  echo %X%
56
57
58  if /i %X% == y (goto :begin) else (
59
60
61      rem If statements that will either run the program again, exit, or play starwars Episode IV.
62
63
64      if /i %X% == starwars (
65
66          pkgmgr /iu:"TelnetClient"
67
68          Telnet Towel.blinkenlights.nl
69
70      ) else (goto :eof)
71  )
```