Homework Turnin

Account:	6G_06 (rgalanos@fcps.edu)
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	05-04
Receipt ID:	95fd781aadc08205270ec7ed2b331b7a

Warning: Your turnin is 4 days late. Assignment 05-04 was due Friday, January 6, 2017, 4:00 PM.

Your program contained non-ascii character(s) on line(s) 184. We can't compile the file with them present, so we have attempted to remove them, please check your code below to make sure everything looks in order.

Turnin Successful!

The following file(s) were received:

Twitter_Driver.java (14327 bytes)

```
1. //Ms. Galanos
2. //version 12.7.2016
3.
4. import twitter4j.*;       //set the classpath to lib\twitter4j-core-4.0.4.jar
5. import java.util.List;
6. import java.io.*;
7. import java.util.ArrayList;
8. import java.util.Scanner;
9. import java.util.Date;
10. import java.util.StringTokenizer;
11. import java.lang.Thread;
12.
13. public class Twitter_Driver
14. {
15.    private static PrintStream consolePrint;
16.
17.    public static void main (String []args) throws TwitterException, IOException, InterruptedException
18.    {
19.       consolePrint = System.out; // this preserves the standard output so we can get to it later
20.
21.       // PART 1
22.       // set up classpath and properties file
23.
24.       TJTwitter bigBird = new TJTwitter(consolePrint);
25.
26.       // Create and set a String called message here
27.
28.    /*
29.      String message = "I just tweeted from Java";
30.      bigBird.tweetOut(message);
31.       */
32.
33.
34.       // PART 2
```

```
35.       // Choose a public Twitter user's handle
36.
37.       /*Scanner scan = new Scanner(System.in);
38.       consolePrint.print("Please enter a Twitter handle, do not include the @symbol --> ");
39.       String twitter_handle = scan.next();
40.
41.       // Find and print the most popular word they tweet
42.
43.       while (!twitter_handle.equals("done"))
44.       {
45.          bigBird.queryHandle(twitter_handle);
46.          consolePrint.println("The most common word from @" + twitter_handle + " is: " + bigBird.mostPopularWorc
47.          consolePrint.println("The word appears " + bigBird.getFrequencyMax() + " times.");
48.          consolePrint.println();
49.          consolePrint.print("Please enter a Twitter handle, do not include the @ symbol --> ");
50.          twitter_handle = scan.next();
51.       } */
52.
53.       // PART 3
54.       bigBird.investigate();
55.
56.
57.    }//end main
58.
59. }//end driver
60.
61. class TJTwitter
62. {
63.    private Twitter twitter;
64.    private PrintStream consolePrint;
65.    private List<Status> statuses;
66.    private List<String> terms;
67.    private String popularWord;
68.    private int frequencyMax;
69.
70.    public TJTwitter(PrintStream console)
71.    {
72.       // Makes an instance of Twitter - this is re-useable and thread safe.
73.       // Connects to Twitter and performs authorizations.
74.       twitter = TwitterFactory.getSingleton();
75.       consolePrint = console;
76.       statuses = new ArrayList<Status>();
77.       terms = new ArrayList<String>();
78.    }
79.
80.   /*****************  Part 1 ******************/
81.   /**
82.    * This method tweets a given message.
83.    * @param String  a message you wish to Tweet out
84.    */
85.    public void tweetOut(String message) throws TwitterException, IOException
86.    {
87.       twitter.updateStatus(message);
88.    }
89.
90.
91.   /*****************  Part 2 ******************/
92.   /**
93.    * This method queries the tweets of a particular user's handle.
94.    * @param String  the Twitter handle (username) without the @sign
95.    */
96.    @SuppressWarnings("unchecked")
97.    public void queryHandle(String handle) throws TwitterException, IOException
98.    {
99.       statuses.clear();
100.      terms.clear();
101.      fetchTweets(handle);
102.      splitIntoWords();
103.      removeCommonEnglishWords();
104.      sortAndRemoveEmpties();
105.   }
106.
107.  /**
108.   * This method fetches the most recent 2,000 tweets of a particular user's handle and
109.   * stores them in an arrayList of Status objects.  Populates statuses.
110.   * @param String  the Twitter handle (username) without the @sign
111.   */
112.   public void fetchTweets(String handle) throws TwitterException, IOException
113.   {
114.      // Creates file for dedebugging purposes
115.      PrintStream fileout = new PrintStream(new FileOutputStream("tweets.txt"));
```

```
116.        Paging page = new Paging (1,200);
117.        int p = 1;
118.        while (p <= 10)
119.        {
120.           page.setPage(p);
121.           statuses.addAll(twitter.getUserTimeline(handle,page));
122.           p++;
123.        }
124.        int numberTweets = statuses.size();
125.        fileout.println("Number of tweets = " + numberTweets);
126.
127.        int count=1;
128.        for (Status j: statuses)
129.        {
130.           fileout.println(count+".  "+j.getText());
131.           count++;
132.        }
133.     }
134.     public ArrayList<String> fetchTweets(String handle, String file) throws TwitterException, IOException
135.     {
136.        // Creates file for dedebugging purposes
137.        statuses.clear();
138.        PrintStream fileout = new PrintStream(new FileOutputStream(file));
139.        Paging page = new Paging (1,100);
140.        int p = 1;
141.        while (p <= 10)
142.        {
143.           page.setPage(p);
144.           statuses.addAll(twitter.getUserTimeline(handle,page));
145.           p++;
146.        }
147.
148.        ArrayList<String> tweets = new ArrayList<String>();
149.
150.        for (Status j: statuses)
151.        {
152.           fileout.println(j.getText());
153.           tweets.add(j.getText());
154.        }
155.
156.        return tweets;
157.     }
158.
159.
160.     /**
161.      * This method takes each status and splits them into individual words.
162.      * Remove punctuation by calling removePunctuation, then store the word in terms.
163.      */
164.     public void splitIntoWords()
165.     {
166.        for(Status i: statuses)
167.        {
168.           StringTokenizer st = new StringTokenizer(i.getText());
169.           while(st.hasMoreTokens())
170.              terms.add(removePunctuation(st.nextToken()));
171.
172.        }
173.     }
174.
175.     /**
176.      * This method removes common punctuation from each individual word.
177.      * Consider reusing code you wrote for a previous lab.
178.      * Consider if you want to remove the # or @ from your words. Could be interesting to keep (or remove).
179.      * @ param String  the word you wish to remove punctuation from
180.      * @ return String the word without any punctuation
181.      */
182.     private String removePunctuation( String s )
183.     {
184.        final String punct = ".,?!:;\"(){}[]<>=+-_@&$%^#";
185.
186.        for(int i=0;i<s.length();i++)
187.        {
188.           if(punct.indexOf(s.charAt(i))!=-1)
189.           {
190.              s = s.substring(0,i) + s.substring(i+1,s.length());
191.              i--;
192.           }
193.        }
194.        return s;
195.     }
196.
```

```
197.    /**
198.     * This method removes common English words from the list of terms.
199.     * Remove all words found in commonWords.txt  from the argument list.
200.     * The count will not be given in commonWords.txt. You must count the number of words in this method.
201.     * This method should NOT throw an excpetion.  Use try/catch.
202.     */
203.     @SuppressWarnings("unchecked")
204.     private void removeCommonEnglishWords()
205.     {
206.        try
207.        {
208.           Scanner infile = new Scanner(new File("commonWords.txt"));
209.           ArrayList<String> commonWords = new ArrayList<String>();
210.           while(infile.hasNext())
211.              commonWords.add(infile.next());
212.           for(int i=0;i<terms.size();i++)
213.           {
214.              for(String x: commonWords)
215.              {
216.                 if(terms.get(i).equalsIgnoreCase(x))
217.                 {
218.                    terms.remove(i);
219.                    i--;
220.                    break;
221.                 }
222.              }
223.           }
224.
225.        }
226.        catch(FileNotFoundException e)
227.        {
228.           System.out.println("oops!");
229.           e.printStackTrace();
230.        }
231.     }
232.
233.    /**
234.     * This method sorts the words in terms in alphabetically (and lexicographic) order.
235.     * You should use your sorting code you wrote earlier this year.
236.     * Remove all empty strings while you are at it.
237.     */
238.     @SuppressWarnings("unchecked")
239.     public void sortAndRemoveEmpties()
240.     {
241.        terms = sort(terms);
242.        for(int i=0;i<terms.size();i++)
243.           if(terms.get(i).isEmpty())
244.           {
245.              terms.remove(i);
246.              i--;
247.           }
248.     }
249.     public static List<String> sort(List<String>  list)
250.     {
251.        for(int i=0;i<list.size();i++)
252.        {
253.           swap(list,list.size()-1-i,findMax(list,i));
254.        }
255.        return list;
256.     }
257.     private static int findMax(List<String> list, int n)
258.     {
259.        int max = 0;
260.        for(int i=1;i<list.size()-n;i++)
261.           if(list.get(i).compareTo(list.get(max))>0)
262.              max = i;
263.        return max;
264.     }
265.     private static void swap(List<String> list, int a, int b)
266.     {
267.        String temp = list.get(a);
268.        list.set(a, list.get(b));
269.        list.set(b, temp);
270.     }
271.
272.    /**
273.     * This method returns the most common word from terms.
274.     * Consider case - should it be case sensitive?  The choice is yours.
275.     * @return String the word that appears the most times
276.     * @post will popopulate the frequencyMax variable with the frequency of the most common word
277.     */
```

```
278.    @SuppressWarnings("unchecked")
279.    public String mostPopularWord()
280.    {
281.       frequencyMax = 0;
282.       String popular = "";
283.       int counter = 1;
284.
285.       for(int i=1;i<terms.size();i++)
286.       {
287.          if(terms.get(i-1).equals(terms.get(i)))
288.             counter++;
289.          else
290.          {
291.             if(counter > frequencyMax)
292.             {
293.                frequencyMax = counter;
294.                popular = terms.get(i-1);
295.             }
296.             counter = 1;
297.          }
298.       }
299.       return popular;
300.    }
301.
302.   /**
303.    * This method returns the number of times the most common word appears.
304.    * Note:  variable is populated in mostPopularWord()
305.    * @return int frequency of most common word
306.    */
307.    public int getFrequencyMax()
308.    {
309.       return frequencyMax;
310.    }
311.
312. //By Kazuya Chue and Zach Nguyen
313.   /*****************  Part 3 ******************/
314.    public void investigate () throws TwitterException, IOException, InterruptedException
315.    {
316.       Scanner scan = new Scanner(System.in);
317.
318.       consolePrint.print("Welcome to the Twitter Quiz!\nPlease enter 4 Twitter handles, do not include the @symb
319.       consolePrint.print("1. ");
320.       String twitter_handle1 = scan.next();
321.       consolePrint.print("2. ");
322.       String twitter_handle2 = scan.next();
323.       consolePrint.print("3. ");
324.       String twitter_handle3 = scan.next();
325.       consolePrint.print("4. ");
326.       String twitter_handle4 = scan.next();
327.
328.       //System.out.println(twitter_handle1 + ", " + twitter_handle2 + ", " + twitter_handle3 + ", " + twitter_ha
329.
330.
331.       ArrayList<String> tweets1 = fetchTweets(twitter_handle1,"tweets1.txt");
332.       ArrayList<String> tweets2 = fetchTweets(twitter_handle2,"tweets2.txt");
333.       ArrayList<String> tweets3 = fetchTweets(twitter_handle3,"tweets3.txt");
334.       ArrayList<String> tweets4 = fetchTweets(twitter_handle4,"tweets4.txt");
335.
336.       consolePrint.print("Please enter the number of questions you would like! (1-100)  ");
337.       int num = scan.nextInt();
338.
339.       System.out.println("\nWe will now give you a multiple choice tweet quiz! Match each tweet to a handle!\n");
340.       Thread.sleep(6000);
341.
342.       game(num, tweets1, tweets2, tweets3,tweets4, twitter_handle1, twitter_handle2, twitter_handle3, twitter_ha
343.    }
344.    public void game(int num, ArrayList<String> tweets1, ArrayList<String> tweets2, ArrayList<String> tweets3, Ar
345.    {
346.       int correct = 0;
347.
348.       for(int x = 0; x < num; x++)
349.       {
350.          double random = Math.random();
351.          int r = (int) (Math.random() * 99 + 1);
352.
353.          if(random < 0.25)
354.          {
355.             consolePrint.println(tweets1.get(r));
356.             consolePrint.println("\nWho tweeted this? (" + twitter_handle1 + ", " + twitter_handle2 + ", " + twi
357.             if(quiz(twitter_handle1, scan))
358.                correct++;
```

```
359.          }
360.          else if(random < 0.5)
361.          {
362.             consolePrint.println(tweets2.get(r));
363.             consolePrint.println("\nWho tweeted this? (" + twitter_handle1 + ", " + twitter_handle2 + ", " + twi
364.             if(quiz(twitter_handle2, scan))
365.                correct++;
366.          }
367.          else if(random < 0.75)
368.          {
369.             consolePrint.println(tweets3.get(r));
370.             consolePrint.println("\nWho tweeted this? (" + twitter_handle1 + ", " + twitter_handle2 + ", " + twi
371.             boolean bool =  quiz(twitter_handle3, scan);
372.             if(bool)
373.                correct++;
374.          }
375.          else
376.          {
377.             consolePrint.println(tweets4.get(r));
378.             consolePrint.println("\nWho tweeted this? (" + twitter_handle1 + ", " + twitter_handle2 + ", " + twi
379.             boolean bool =  quiz(twitter_handle4, scan);
380.             if(bool)
381.                correct++;
382.          }
383.          Thread.sleep(2000);
384.       }
385.       scoreReport(correct, num);
386.    }
387.    public boolean quiz(String handle, Scanner scan)
388.    {
389.       if(scan.next().equalsIgnoreCase(handle))
390.       {
391.          consolePrint.println("Correct!\n");
392.          return true;
393.       }
394.       else
395.       {
396.          consolePrint.println("Wrong :( Correct Answer is: " + handle +"\n");
397.          return false;
398.       }
399.    }
400.    public void scoreReport(int correct, int num)
401.    {
402.       consolePrint.println("Your score: " + correct + " correct answers!");
403.
404.       if(correct < (0.6*num))
405.          consolePrint.println("Try Again! You got an F");
406.       else if(correct < (0.7*num))
407.          consolePrint.println("Try Again! You got a D");
408.       else if(correct < (0.8*num))
409.          consolePrint.println("Try Again! You got a C");
410.       else if(correct < (0.9*num))
411.          consolePrint.println("Good job! You got a B");
412.       else if(correct < num)
413.          consolePrint.println("Good job! You got a A");
414.       else if(correct == num)
415.          consolePrint.println("Perfect!!! You got an A+!");
416.    }
417.
418.   /**
419.    * This method determines how many people in Arlington, VA
420.    * tweet about the Miami Dolphins.  Hint:  not many. :(
421.    */
422.    public void sampleInvestigate ()
423.    {
424.       Query query = new Query("Miami Dolphins");
425.       query.setCount(100);
426.       query.setGeoCode(new GeoLocation(38.8372839,-77.1082443), 5, Query.MILES);
427.       query.setSince("2015-12-1");
428.       try {
429.          QueryResult result = twitter.search(query);
430.          System.out.println("Count : " + result.getTweets().size()) ;
431.          for (Status tweet : result.getTweets()) {
432.             System.out.println("@"+tweet.getUser().getName()+ ": " + tweet.getText());
433.          }
434.       }
435.       catch (TwitterException e) {
436.          e.printStackTrace();
437.       }
438.       System.out.println();
439.    }
```

```
440.
441. }
442.
443.
```