# Homework Turnin

| | |
|---|---|
| **Email:** | rgalanos@fcps.edu |
| **Section:** | 6G |
| **Course:** | TJHSST APCS 2016-17 |
| **Assignment:** | 05-03 |
| **Receipt ID:** | 6a8a523e240b176737096d98b5ddf33a |

Execution failed with return code 1 (general error). (Expected for JUnit when any test fails.)

**Warning**: Your program failed to compile :

```
Twitter_Driver.java:4: error: package twitter4j does not exist
import twitter4j.*;       //set the classpath to lib\twitter4j-core-4.0.4.jar
^
Twitter_Driver.java:16: error: cannot find symbol
   public static void main (String []args) throws TwitterException, IOException
                                                   ^
  symbol:   class TwitterException
  location: class Twitter_Driver
Twitter_Driver.java:62: error: cannot find symbol
   private Twitter twitter;
           ^
  symbol:   class Twitter
  location: class TJTwitter
Twitter_Driver.java:64: error: cannot find symbol
   private List<Status> statuses;
               ^
  symbol:   class Status
  location: class TJTwitter
Twitter_Driver.java:84: error: cannot find symbol
   public void tweetOut(String message) throws TwitterException, IOException
                                               ^
  symbol:   class TwitterException
  location: class TJTwitter
Twitter_Driver.java:96: error: cannot find symbol
   public void queryHandle(String handle) throws TwitterException, IOException
                                                  ^
  symbol:   class TwitterException
  location: class TJTwitter
Twitter_Driver.java:111: error: cannot find symbol
   public void fetchTweets(String handle) throws TwitterException, IOException
                                                  ^
  symbol:   class TwitterException
  location: class TJTwitter
```

```
Twitter_Driver.java:73: error: cannot find symbol
      twitter = TwitterFactory.getSingleton();
                ^
  symbol:   variable TwitterFactory
  location: class TJTwitter
Twitter_Driver.java:75: error: cannot find symbol
      statuses = new ArrayList<Status>();
                               ^
  symbol:   class Status
  location: class TJTwitter
Twitter_Driver.java:115: error: cannot find symbol
      Paging page = new Paging (1,200);
      ^
  symbol:   class Paging
  location: class TJTwitter
Twitter_Driver.java:115: error: cannot find symbol
      Paging page = new Paging (1,200);
                        ^
  symbol:   class Paging
  location: class TJTwitter
Twitter_Driver.java:127: error: cannot find symbol
      for (Status j: statuses)
           ^
  symbol:   class Status
  location: class TJTwitter
Twitter_Driver.java:140: error: cannot find symbol
      for(Status i: statuses)
          ^
  symbol:   class Status
  location: class TJTwitter
Twitter_Driver.java:299: error: cannot find symbol
      Query query = new Query("Miami Dolphins");
      ^
  symbol:   class Query
  location: class TJTwitter
Twitter_Driver.java:299: error: cannot find symbol
      Query query = new Query("Miami Dolphins");
                        ^
  symbol:   class Query
  location: class TJTwitter
Twitter_Driver.java:301: error: cannot find symbol
      query.setGeoCode(new GeoLocation(38.8372839,-77.1082443), 5, Query.MILES);
                           ^
  symbol:   class GeoLocation
  location: class TJTwitter
Twitter_Driver.java:301: error: cannot find symbol
      query.setGeoCode(new GeoLocation(38.8372839,-77.1082443), 5, Query.MILES);
                                                                   ^
  symbol:   variable Query
  location: class TJTwitter
Twitter_Driver.java:304: error: cannot find symbol
        QueryResult result = twitter.search(query);
```

```
          ^
  symbol:   class QueryResult
  location: class TJTwitter
Twitter_Driver.java:306: error: cannot find symbol
         for (Status tweet : result.getTweets()) {
              ^
  symbol:   class Status
  location: class TJTwitter
Twitter_Driver.java:310: error: cannot find symbol
      catch (TwitterException e) {
             ^
  symbol:   class TwitterException
  location: class TJTwitter
20 errors
```

---

Please correct your file(s), go back, and try to submit again. If you do not correct this problem, you are likely to lose a large number of points on the assignment. Please **contact your TA** if you are not sure why your code is not compiling successfully.

# Turnin Failed! (See above)

There were some problems with your turnin. Please look at the messages above, fix the problems, then Go Back and try your turnin again.

GradeIt has a copy of your submission, but we believe that you will want to fix the problems with your submission by resubmitting a fixed version of your code by the due date.

We have received the following file(s):

## Twitter_Driver.java (9270 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.
12. public class Twitter_Driver
13. {
14.    private static PrintStream consolePrint;
15.
16.    public static void main (String []args) throws TwitterException, IOException
17.    {
18.       consolePrint = System.out; // this preserves the standard output so we can get to it later
19.
20.       // PART 1
21.       // set up classpath and properties file
22.
23.       TJTwitter bigBird = new TJTwitter(consolePrint);
```

```
24.
25.       // Create and set a String called message here
26.
27.    /*
28.      String message = "I just tweeted from Java";
29.      bigBird.tweetOut(message);
30.        */
31.
32.
33.       // PART 2
34.       // Choose a public Twitter user's handle
35.
36.       Scanner scan = new Scanner(System.in);
37.       consolePrint.print("Please enter a Twitter handle, do not include the @symbol --> ");
38.       String twitter_handle = scan.next();
39.
40.       // Find and print the most popular word they tweet
41.
42.       while (!twitter_handle.equals("done"))
43.       {
44.          bigBird.queryHandle(twitter_handle);
45.          consolePrint.println("The most common word from @" + twitter_handle + " is: " + bigBird.mostPopularWord()
46.          consolePrint.println("The word appears " + bigBird.getFrequencyMax() + " times.");
47.          consolePrint.println();
48.          consolePrint.print("Please enter a Twitter handle, do not include the @ symbol --> ");
49.          twitter_handle = scan.next();
50.       }
51.
52.       // PART 3
53.       //bigBird.investigate();
54.
55.
56.    }//end main
57.
58. }//end driver
59.
60. class TJTwitter
61. {
62.    private Twitter twitter;
63.    private PrintStream consolePrint;
64.    private List<Status> statuses;
65.    private List<String> terms;
66.    private String popularWord;
67.    private int frequencyMax;
68.
69.    public TJTwitter(PrintStream console)
70.    {
71.       // Makes an instance of Twitter - this is re-useable and thread safe.
72.       // Connects to Twitter and performs authorizations.
73.       twitter = TwitterFactory.getSingleton();
74.       consolePrint = console;
75.       statuses = new ArrayList<Status>();
76.       terms = new ArrayList<String>();
77.    }
78.
79.   /*****************  Part 1 ******************/
80.   /**
81.    * This method tweets a given message.
82.    * @param String  a message you wish to Tweet out
83.    */
84.    public void tweetOut(String message) throws TwitterException, IOException
85.    {
86.       twitter.updateStatus(message);
87.    }
88.
89.
90.   /*****************  Part 2 ******************/
91.   /**
92.    * This method queries the tweets of a particular user's handle.
93.    * @param String  the Twitter handle (username) without the @sign
94.    */
95.    @SuppressWarnings("unchecked")
96.    public void queryHandle(String handle) throws TwitterException, IOException
97.    {
98.       statuses.clear();
99.       terms.clear();
100.      fetchTweets(handle);
101.      splitIntoWords();
102.      removeCommonEnglishWords();
103.      sortAndRemoveEmpties();
104.   }
105.
```

```
106.    /**
107.     * This method fetches the most recent 2,000 tweets of a particular user's handle and
108.     * stores them in an arrayList of Status objects.  Populates statuses.
109.     * @param String  the Twitter handle (username) without the @sign
110.     */
111.    public void fetchTweets(String handle) throws TwitterException, IOException
112.    {
113.       // Creates file for dedebugging purposes
114.       PrintStream fileout = new PrintStream(new FileOutputStream("tweets.txt"));
115.       Paging page = new Paging (1,200);
116.       int p = 1;
117.       while (p <= 10)
118.       {
119.          page.setPage(p);
120.          statuses.addAll(twitter.getUserTimeline(handle,page));
121.          p++;
122.       }
123.       int numberTweets = statuses.size();
124.       fileout.println("Number of tweets = " + numberTweets);
125.
126.       int count=1;
127.       for (Status j: statuses)
128.       {
129.          fileout.println(count+".  "+j.getText());
130.          count++;
131.       }
132.    }
133.
134.    /**
135.     * This method takes each status and splits them into individual words.
136.     * Remove punctuation by calling removePunctuation, then store the word in terms.
137.     */
138.    public void splitIntoWords()
139.    {
140.       for(Status i: statuses)
141.       {
142.          StringTokenizer st = new StringTokenizer(i.getText());
143.          while(st.hasMoreTokens())
144.             terms.add(removePunctuation(st.nextToken()));
145.
146.       }
147.    }
148.
149.    /**
150.     * This method removes common punctuation from each individual word.
151.     * Consider reusing code you wrote for a previous lab.
152.     * Consider if you want to remove the # or @ from your words. Could be interesting to keep (or remove).
153.     * @ param String  the word you wish to remove punctuation from
154.     * @ return String the word without any punctuation
155.     */
156.    private String removePunctuation( String s )
157.    {
158.       final String punct = ".,?!:;\"(){}[]<>=+-_@&$%^#";
159.
160.       for(int i=0;i<s.length();i++)
161.       {
162.          if(punct.indexOf(s.charAt(i))!=-1)
163.          {
164.             s = s.substring(0,i) + s.substring(i+1,s.length());
165.             i--;
166.          }
167.       }
168.       return s;
169.    }
170.
171.    /**
172.     * This method removes common English words from the list of terms.
173.     * Remove all words found in commonWords.txt  from the argument list.
174.     * The count will not be given in commonWords.txt. You must count the number of words in this method.
175.     * This method should NOT throw an excpetion.  Use try/catch.
176.     */
177.    @SuppressWarnings("unchecked")
178.    private void removeCommonEnglishWords()
179.    {
180.       try
181.       {
182.          Scanner infile = new Scanner(new File("commonWords.txt"));
183.          ArrayList<String> commonWords = new ArrayList<String>();
184.          while(infile.hasNext())
185.             commonWords.add(infile.next());
186.          for(int i=0;i<terms.size();i++)
187.          {
```

```
            for(String x: commonWords)
            {
               if(terms.get(i).equalsIgnoreCase(x))
               {
                  terms.remove(i);
                  i--;
                  break;
               }
            }
         }

      }
      catch(FileNotFoundException e)
      {
         System.out.println("oops!");
         e.printStackTrace();
      }
   }

   /**
    * This method sorts the words in terms in alphabetically (and lexicographic) order.
    * You should use your sorting code you wrote earlier this year.
    * Remove all empty strings while you are at it.
    */
   @SuppressWarnings("unchecked")
   public void sortAndRemoveEmpties()
   {
      terms = sort(terms);
      for(int i=0;i<terms.size();i++)
         if(terms.get(i).isEmpty())
         {
            terms.remove(i);
            i--;
         }
   }
   public static List<String> sort(List<String>  list)
   {
      for(int i=0;i<list.size();i++)
      {
         swap(list,list.size()-1-i,findMax(list,i));
      }
      return list;
   }
   private static int findMax(List<String> list, int n)
   {
      int max = 0;
      for(int i=1;i<list.size()-n;i++)
         if(list.get(i).compareTo(list.get(max))>0)
            max = i;
      return max;
   }
   private static void swap(List<String> list, int a, int b)
   {
      String temp = list.get(a);
      list.set(a, list.get(b));
      list.set(b, temp);
   }

   /**
    * This method returns the most common word from terms.
    * Consider case - should it be case sensitive?  The choice is yours.
    * @return String the word that appears the most times
    * @post will popopulate the frequencyMax variable with the frequency of the most common word
    */
   @SuppressWarnings("unchecked")
   public String mostPopularWord()
   {
      frequencyMax = 0;
      String popular = "";
      int counter = 1;

      for(int i=1;i<terms.size();i++)
      {
         if(terms.get(i-1).equals(terms.get(i)))
            counter++;
         else
         {
            if(counter > frequencyMax)
            {
               frequencyMax = counter;
               popular = terms.get(i-1);
            }
```

```
            counter = 1;
         }
      }
      return popular;
   }

   /**
    * This method returns the number of times the most common word appears.
    * Note:  variable is populated in mostPopularWord()
    * @return int frequency of most common word
    */
   public int getFrequencyMax()
   {
      return frequencyMax;
   }


   /*****************  Part 3 ******************/
   public void investigate ()
   {
      //Enter your code here
   }

   /**
    * This method determines how many people in Arlington, VA
    * tweet about the Miami Dolphins.  Hint:  not many. :(
    */
   public void sampleInvestigate ()
   {
      Query query = new Query("Miami Dolphins");
      query.setCount(100);
      query.setGeoCode(new GeoLocation(38.8372839,-77.1082443), 5, Query.MILES);
      query.setSince("2015-12-1");
      try {
         QueryResult result = twitter.search(query);
         System.out.println("Count : " + result.getTweets().size()) ;
         for (Status tweet : result.getTweets()) {
            System.out.println("@"+tweet.getUser().getName()+ ": " + tweet.getText());
         }
      }
      catch (TwitterException e) {
         e.printStackTrace();
      }
      System.out.println();
   }

}
```