APP.java

```java
package app;

/**
 *
 * <p><strong><em>Application Name: </em></strong>Program 3, Spell Checker</p>
 * <p><strong><em>Class Name: </em></strong>App</p>
 *
 * <p><strong><em>Application Notes: </em></strong>none</p>
 *
 * <p><strong><em>Class Notes: </em></strong>none</p>
 *
 * <p><strong><em>Pre-Conditions: </em></strong>oliver.txt file must exist. dictionary.txt file must exist</p>
 *
 * <p><strong><em>Post-Conditions: </em></strong>none</p>
 *
 * <p><strong><em>Author: </em></strong>Daniel C. Landon Jr.</p>
 * <p><strong><em>Instructor: </em></strong>Dr. Robert Walsh</p>
 * <p><strong><em>Course: </em></strong>SP20-SE-CSCI-C202-17057</p>
 * <p><strong><em>Due Date: </em></strong>03.26.2020</p>
 *
 */
public class App {

    /**
     *
     * <p><strong><em>Description: </em></strong>application entry point</p>
     *
     * <p><strong><em>Method Name: </em></strong>main</p>
     *
     * <p><strong><em>Method Notes: </em></strong>none</p>
     *
     * <p><strong><em>Pre-Conditions: </em></strong>none</p>
     *
     * <p><strong><em>Post-Conditions: </em></strong>none</p>
     *
     * <p><strong><em>Author: </em></strong>Daniel C. Landon Jr.</p>
     * <p><strong><em>Start Date: </em></strong>03.25.2020</p>
     *
     * @param args not used
     * @throws Exception not used
     */
    public static void main(String[] args) throws Exception {
        ReadFile.countSpellingErrors();
    }
}
```

ReadFile.java

```
package app;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

/**
 *
 * <p>
 * <strong><em>Class Name: </em></strong>ReadFile
 * </p>
 *
 * <p>
 * <strong><em>Application Notes: </em></strong>none
 * </p>
 *
 * <p>
 * <strong><em>Class Notes: </em></strong>none
 * </p>
 *
 * <p>
 * <strong><em>Pre-Conditions: </em></strong>none
 * </p>
 *
 * <p>
 * <strong><em>Post-Conditions: </em></strong>none
 * </p>
 *
 * <p>
 * <strong><em>Author: </em></strong>Daniel C. Landon Jr.
 * </p>
 * <p>
 * <strong><em>Instructor: </em></strong>Dr. Robert Walsh
 * </p>
 * <p>
 * <strong><em>Course: </em></strong>SP20-SE-CSCI-C202-17057
 * </p>
 * <p>
 * <strong><em>Start Date: </em></strong>03.26.2020
 * </p>
 * <p>
 * <strong><em>Due Date: </em></strong>03.26.2020
 * </p>
 *
 */
public class ReadFile {

    public static File _dictionary = new File("dictionary.txt");
public static File _oliver = new File("oliver.txt");
public static int _wordCount = 0;
public static int _correctWords = 0;
public static int _misspelledWords = 0;
public static final int DICTIONARYLENGTH = 235887;

/**
 *
 * @throws FileNotFoundException
 * @throws Exception
 */
public static void countSpellingErrors() throws FileNotFoundException, Exception{
    Scanner _file = new Scanner(_oliver); //read from oliver.txt

    while(_file.hasNextLine()){

        String[] nextLine = formatLine(_file.nextLine());

        for (int i = 0; i < nextLine.length; i++) {

            _wordCount++;
            // if(_wordCount % 500 == 0){System.out.println("word number: " + _wordCount); }// ECHO

            // if(recursiveBinarySearch(dictionary(),nextLine[i]) == -1) { _misspelledWords++; } // end if
            // else{ _correctWords++; } // end else
        } // end for i

    } // end while

    System.out.println("Misspelled words: " + _misspelledWords);
    System.out.println("Correct words: " + _correctWords);
    System.out.println("Total words: " + _wordCount);

    _file.close();

} // end countSpellingErrors

/**
 *
 * @return array
 * @throws FileNotFoundException
 * @throws Exception
 */
public static String[] dictionary() throws FileNotFoundException, Exception{

    Scanner dictionaryFile = new Scanner(_dictionary);//read from dictionary.txt
    String[] dictionaryArray = new String[DICTIONARYLENGTH];//array to hold all words in dictionary
    int dictionaryEntryNumber = 0;//index at which to add word from dictionaryFile

    while(dictionaryFile.hasNextLine()){

        dictionaryArray[dictionaryEntryNumber] = dictionaryFile.nextLine();
```

```
            dictionaryEntryNumber++;

        } // end while

        dictionaryFile.close();

        return dictionaryArray;

    } // end dictionary

    /**
     *
     * @param line what to process
     * @return array
     */
    public static String[] formatLine(String line){

        String str = line.replaceAll("'", ""); // removes all apostrophes

        str = str.toLowerCase();
        str = str.replaceAll("[^a-zA-Z\\s]", " ").trim(); // replaces all non-alpha and non-
space characters with a space
        str = str.replaceAll("\\s+", " "); // replaces all double or more spaces with one space

        String[] outputArray = str.split(" ");

        return outputArray;

    } // end formatLine

    /**
     *
     * @param array what to process
     * @param key key for keeping track
     * @return recursion
     */
    public static int recursiveBinarySearch(String[] array, String key){

        int low = 0;
        int high = array.length - 1;

        return helperBinarySearch(array, key, low, high);

    } // end recursiveBinarySearch

    /**
     *
     * @param array what to process
     * @param key key value
     * @param low low point
     * @param high high point
     * @return found or not
     */
    private static int helperBinarySearch(String[] array, String key, int low, int high){

        if (low > high){ return -1; } // if key is not found in list

        int mid = (low + high) / 2;

        if(array[mid].compareToIgnoreCase(key) == 0) { return mid; } // if key found at array[mid]

        else if (array[mid].compareToIgnoreCase(key) > 0) { return helperBinarySearch(array, key, low, mid - 1);
}//else if key is lower alphabetically than array[mid]

        else { return helperBinarySearch(array, key, mid + 1, high); } //else - key is higher alphabetically than arra
y[mid]

    } // end helperBinarySearch

} // end ReadFile
```

Console Output

```
Misspelled words: 0
Correct words: 0
Total words: 1004317
```