```java
/* SELF ASSESSMENT

1. readDictionary
- I have the correct method definition [Mark out of 5:5]
- Comment: My method definition was correct.
- My method reads the words from the "words.txt" file. [Mark out of 5:5]
- Comment: It reads from the word.txt file.
- It returns the contents from "words.txt" in a String array or an ArrayList. [Mark out of 5:5]
- Comment: Reads it to an String Array

2. readWordList
- I have the correct method definition [Mark out of 5:5]
- Comment: I used the correct method definition.
- My method reads the words provided (which are separated by commas, saves them to an array or ArrayList of String references and returns it.
[Mark out of 5:5]
- Comment: It reads the words given and seperated them by commas and then saves them to an String Array.

3. isUniqueList
- I have the correct method definition [Mark out of 5:5]
- Comment: I used the correct method definition.
- My method compares each word in the array with the rest of the words in the list. [Mark out of 5:5]
- Comment: My method does the above.
- Exits the loop when a non-unique word is found. [Mark out of 5:5]
- Comment: If this hapeens it will return false.
- Returns true is all the words are unique and false otherwise. [Mark out of 5:5]
- Comment: My method does the above.

4. isEnglishWord
- I have the correct method definition [Mark out of 5:5]
- Comment: I used the correct method defintion.
- My method uses the binarySearch method in Arrays library class. [Mark out of 3:3]
- Comment: My method does the above.
- Returns true if the binarySearch method return a value >= 0, otherwise false is returned. [Mark out of 2:2]
- Comment: It does the above succesfully.

5. isDifferentByOne
- I have the correct method definition [Mark out of 5:5]
- Comment: I used the correct method definition.
- My method loops through the length of a words comparing characters at the same position in both words searching for one difference. [Mark
out of 10:10]
- Comment: My method does the above.

6. isWordChain
- I have the correct method definition [Mark out of 5:5]
- Comment: I used the correct method definition.
- My method calls isUniqueList, isEnglishWord and isDifferentByOne methods and prints the appropriate message [Mark out of 10:10]
- Comment: My method calls all of the above and prints the appropriate message.

7. main
- Reads all the words from file words.txt into an array or an ArrayList using the any of teh Java.IO classes covered in lectures [Mark out of
10:10]
- Comment: My mains does the above.
- Asks the user for input and calls isWordChain [Mark out of 5:5]
- Comment: Does the above.

 Total Mark out of 100 (Add all the previous marks):100
*/

import java.util.Scanner;
import java.util.ArrayList;
import java.util.Locale;
import java.io.*;
import java.util.Arrays;

public class LewisCarrolsWordLinksPuzzleGame {

	public static String[] readDictionary()
	{
		String[] dictionaryOfWords = new String[658964];
		try {
			FileReader fileReader = new FileReader("words.txt");
			BufferedReader bufferedReader = new BufferedReader(fileReader);
			String reader;
			int i = 0;
			while((reader = bufferedReader.readLine()) != null) {
				dictionaryOfWords[i] = reader;
				i += 1;
			}
			bufferedReader.close();
			fileReader.close();
		}catch (FileNotFoundException e)	{
			e.printStackTrace();
		}catch (IOException e)	{
			 e.printStackTrace();
		}
		return dictionaryOfWords;
```

```
    }

    public static String[] readWordList(String userInput)
    {
            String[] listOfInputWords = userInput.split(",");

            return listOfInputWords;
    }

    public static boolean isUniqueList(String[] listOfInputWords)
    {
            boolean allUnique = true;
            for(int count1 = 0; count1<listOfInputWords.length;count1++)
            {
                    for(int count2 = 0;count2<listOfInputWords.length;count2++)
                    {
                            if(count1!=count2)
                            {
                                    if(listOfInputWords[count1]==listOfInputWords[count2]) allUnique=false;
                            }
                    }
            }
            return allUnique;
    }

    public static boolean isEnglishWord(String word, String[] dictionaryOfWords)
    {
            if (Arrays.binarySearch(dictionaryOfWords, word) > 0)
            {
                    return true;
            }
            return false;
    }

    public static boolean isDifferentByOne(String[] listOfInputWords)
    {
            for(int count = 0; count < listOfInputWords.length - 1; count++)
            {
                    int numberOfDiffrences=0;
                    char[] firstArray = listOfInputWords[count].toCharArray();
                    char[] secondArray = listOfInputWords[count+1].toCharArray();
                    if(firstArray.length==secondArray.length)
                    {
                            for(int counter =0;counter<firstArray.length;counter++)
                            {
                                    if(firstArray[counter]!=secondArray[counter]) numberOfDiffrences++;
                            }
                            if(numberOfDiffrences!=1) return false;
                    }
                    else
                    {
                            return false;
                    }
            }
            return true;
    }

    public static boolean isWordChain(String[] listOfInputWords, String[] dictionaryOfWords)
    {
            boolean isUniqueList = isUniqueList(listOfInputWords);
            if(isUniqueList)
            {
                    boolean isDiffrentByOne = isDifferentByOne(listOfInputWords);
                    if(isDiffrentByOne)
                    {
                            for(int count = 0;count<listOfInputWords.length; count++)
                            {
                                    String word = listOfInputWords[count];
                                    boolean isEnglishWord = isEnglishWord(word, dictionaryOfWords);
                                    if(!isEnglishWord) return false;
                            }
                            return true;
                    }
            }
            return false;
    }

    public static void main(String[] args) {

            Scanner input = new Scanner( System.in );

            boolean finished = false;
            String[] dictionaryOfWords = readDictionary();
            while(!finished)
            {
                    System.out.println("Enter a comma separated list of words (or an empty list to quit):\n");
				String userInput = input.next();
				if(!userInput.equals(""))
				{
					String[] listOfInputWords = readWordList(userInput);
					boolean isWordChain = isWordChain(listOfInputWords, dictionaryOfWords);
					if(isWordChain) System.out.println("\n\nValid chain of words from Lewis Carroll's word-links game.\n");
					else System.out.println("\n\nNot a valid chain of words from Lewis Carroll's word-links game.\n");
				}
				else
				{
					finished=true;
				}
			}
			System.out.println("Quit Program Succesful.");
			input.close();
		}

}
```