Coding Challenge Requirement:

1. Read input from a file of words;

2. Find the largest word in the file

3. Transpose the letters in the largest word

4. Show the largest word and the largest word transposed

5. Please use automated test cases using a test framework

6. Demonstrate positive and negative test cases

7. Your challenge should allow the user to pass in a file

8. Ensure you document code and instructions for building and running based on the response best practices above

Assumptions:

* Assuming the input file format is text file only.
* Assuming the file is not having two longest words (i.e duplicate words or two words having same length).
* If the file is having two longest words, then print the first occurrence of longest word.
* Ignoring the blank and white spaces in a word.
* Considering the word can be combination with numbers, special characters or junk characters as well

Design:

1.Class -TransposeLongestWord:

Methods:

public static String longestWord(String Filepath) : Read the file and return the longest word

public static String TransposeWord(String longestword): Reverse the given string and return it.

public static void main(String [] args) ; To run or test the above methods with user interaction

2.Class - JunitTest: Created a class using Junit testing framework to test the above methods with different sets of test data files

Positive Tests:

* public void testvalidfile\_first()
* public void testvalidfile\_middle()
* public void testvalidfile\_last()
* public void testvalidfile\_numbersonlyword()
* public void testvalidfile\_charsnumswords()
* public void testvalidfile\_specialcharsword()

Negative Tests:

* public void testInvalidfileformat()
* public void testfilenotfound()
* public void testEmptyfile()

3. Class- TestRunner: This class is used to run the JunitTest Test methods in the command prompt

Code:

Create a project in Eclipse or any JDK environment with below 3 class files or import the files from github once its downloaded and add the downloaded Junit jars to the project . Or Also simply create 3 text files with below three class files and saved the file name as <classname>.java .

Download the below files from Github link: <https://github.com/raju1284/CodeChallenge.git>

**1.Class -TransposeLongestWord:**

import java.io.\*;

import java.util.\*;

public class TransposeLongestWord {

// Method to read the input file and find the longest word and return the word

public static String longestWord(String Filepath) throws FileNotFoundException

{

String longest\_word = "";

String current;

//Checking the file format

if (Filepath.endsWith(".txt"))

{

try {

//Reading the file to scanner class

Scanner sc = new Scanner(new File(Filepath));

//checking the longest word

while (sc.hasNext()) {

current = sc.next();

if (current.length() > longest\_word.length()) {

longest\_word = current;

}

}

}

//Handling the File not found exception

catch(Exception FileNotFoundException)

{

longest\_word = "File not Found";

}

}

else

longest\_word = "Please enter the path in text file format only";

if (longest\_word=="")

longest\_word="Emptyfile No words";

return longest\_word;

}

//Method to reverse a string and return the string using StringBuilder reverse function

public static String TransposeWord(String longestword)

{

StringBuilder Tlongest\_word = new StringBuilder();

Tlongest\_word.append(longestword);

Tlongest\_word.reverse();

return Tlongest\_word.toString();

}

// Main method to test the above methods

public static void main(String [] args) throws FileNotFoundException

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the full path of file with file name and its extension:");

String Filenamepath = sc.nextLine(); // Reading the input file path from the user

String logestword = longestWord(Filenamepath); // finding the longest word

if (logestword=="File not Found" || logestword=="Please enter the path in text file format only")

System.out.println(logestword);

else

{

System.out.println("Longesword in a file:"+logestword);

String transposeword =TransposeWord(logestword); //Reverse the string

System.out.println("Transposed word:"+transposeword);

}

sc.close();

}

}**2.Class - JunitTest:**

import static org.junit.Assert.assertEquals;

import java.io.FileNotFoundException;

import org.junit.Test;

public class JunitTest {

// Initaile the TransposeLongestWord class

TransposeLongestWord TLword = new TransposeLongestWord();

String TestDataFilesPath ="C:\\Users\\"+System.getenv("USERNAME")+"\\Desktop\\CodeChallenge\\TestDataFiles\\";

//Test1- Longest Words is at the begining of the file

@Test

public void testvalidfile\_first() throws FileNotFoundException {

System.out.println("Valid File - Longest word at the beginning file");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath+"Test1.txt");

String transposeword =TransposeLongestWord.TransposeWord(logestword);

assertEquals("abcde",logestword);

assertEquals("edcba",transposeword);

}

//Test2- Longest Word is at the middle of the file

@Test

public void testvalidfile\_middle() throws FileNotFoundException {

System.out.println("Valid File - Longest word at the middle of file");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath+"Test2.txt");

String transposeword =TransposeLongestWord.TransposeWord(logestword);

assertEquals("abcde",logestword);

assertEquals("edcba",transposeword);

}

//Test3- Longest Word is at the End of the file

@Test

public void testvalidfile\_last() throws FileNotFoundException {

System.out.println("Valid File - Longest word at the endfile");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath+"Test3.txt");

String transposeword =TransposeLongestWord.TransposeWord(logestword);

assertEquals("abcde",logestword);

assertEquals("edcba",transposeword);

}

//Test4- Longest Word is in numeric format

@Test

public void testvalidfile\_numbersonlyword() throws FileNotFoundException {

System.out.println("Valid File - Longest word is numbers only");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath +"Test4.txt");

String transposeword =TransposeLongestWord.TransposeWord(logestword);

assertEquals("12345",logestword);

assertEquals("54321",transposeword);

}

//Test5- Longest Word is in AlphaNumeric format

@Test

public void testvalidfile\_charsnumswords() throws FileNotFoundException {

System.out.println("Valid File - Longest word is alphanumeric");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath +"Test5.txt");

String transposeword =TransposeLongestWord.TransposeWord(logestword);

assertEquals("abc123",logestword);

assertEquals("321cba",transposeword);

}

//Test6- Longest Word is in with special chars format

@Test

public void testvalidfile\_specialcharsword() throws FileNotFoundException {

System.out.println("Valid File - Longest word is included with special chars");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath +"Test6.txt");

String transposeword =TransposeLongestWord.TransposeWord(logestword);

assertEquals("abcde$%",logestword);

assertEquals("%$edcba",transposeword);

}

//Test7- Invalid File type

@Test

public void testInvalidfileformat() throws FileNotFoundException {

System.out.println("Incorrect File extension");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath +"Test1.xls");

assertEquals("Please enter the path in text file format only",logestword);

}

//Test8- Given File not found

@Test

public void testfilenotfound() throws FileNotFoundException {

System.out.println("File not found");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath +"Test9.txt");

assertEquals("File not Found",logestword);

}

//Test9- Empty File No Data or Spaces

@Test

public void testEmptyfile() throws FileNotFoundException {

System.out.println("Empty File");

String logestword = TransposeLongestWord.longestWord(TestDataFilesPath +"Emptyfile.txt");

assertEquals("Emptyfile No words",logestword);

}

}

**3.Class- TestRunner:**

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(JunitTest.class);

System.out.printf("Test ran: %s, Failed: %s%n",

result.getRunCount(), result.getFailureCount());

for (Failure failure : result.getFailures()) {

System.out.println(failure.toString());

}

// final result, pass or fail

System.out.println("All test cases are passed? " + result.wasSuccessful());

}

}

**Execute/Run:**  Provide steps how to run these classes using command prompt in another ReadMe document with required commands and screen prints.