The University of the West Indies, St. Augustine
COMP 3607 Object Oriented Programming II
2023/2024 Semester 1
Assignment 2

Release Date: October 26, 2023
Due Date: November 04, 2023 at 10:00 p.m.
Bonus Days Allowed: 4

Problem Scenario:

You are tasked with writing a Java program that evaluates the similarity of two pieces of text using different techniques for two tasks: spell checking and lesson recommendation. Three popular text similarity metrics are to be used: (1) Cosine similarity, (2) Levenshtein distance, and (3) Jaccard coefficient of similarity.

Use the **Template** design pattern in your solution where your base class should be called **TextSimilarityChecker** with one public template method called **evaluate**(String filename1, String filename2) that returns the similarity score as a double value. It should use at least one abstract method and one concrete method in the steps. The TextSimilarityChecker is specialised by the **SpellChecker** and **LessonRecommender** classes. The former uses (2) as its default similarity metric whereas the latter uses (3) as its default similarity metric. Both have overloaded constructors that allow selection of different metrics as needed. See code snippet below for examples.

Use the **Strategy** design pattern to implement the algorithms where a **SimilarityMetric** interface declares a **measure(String s1, String s2)** method that each of the algorithmic subtype defines.

The following links provide implementations of the algorithms, which you may have to refactor:
http://www.java2s.com/example/java-utility-method/cosine-similarity/cosinesimilarity-string-tkn0-string-tkn1-a50fe.html

http://www.java2s.com/example/java-utility-method/string-levenshtein-distance/levenshtein-string-s-string-t-b58ae.html

http://www.java2s.com/example/java/java.lang/compute-jaccards-coefficient-of-similarity.html

Useful background reading link:https://spotintelligence.com/2022/12/19/text-similarity-python/

Therefore, given two files containing text data, any of the algorithms can be invoked in **App.java (main class)** using polymorphic object creation as follows:

```
ArrayList<TextSimilarityChecker> checkers = new ArrayList<TextSimilarityChecker>( );
checkers.add( new SpellChecker( ) ); // uses Levenshtein as default
checkers.add( new SpellChecker(1) ); // uses Cosine
checkers.add( new SpellChecker(3) ); // uses Jaccard
checkers.add( new LessonRecommender( ) ); // uses Cosine as default
checkers.add( new LessonRecommender(3); // uses Jaccard
checkers.add( new LessonRecommender(2); // uses Levenshtein
for(TextSimilarityChecker checker: checkers)
        System.out.println( checker.evaluate("file1.txt", "file2.txt") );
```

Build your solution as a Maven project and name your package **comp3607A2**

Submission Instructions

- Ensure your student ID is documented in your submission.
- State any assumptions that you make. Cite any sources used as appropriate.
- Label all answers and diagrams appropriately based on the question part answered.
- Upload your Maven submission as a zipped file to the myeLearning course page by the deadline.
- If you are using your bonus day(s), please indicate on your submission within the text area of the submission portal.
- Sign and include the plagiarism declaration form. This is an individual assignment that should be completed on your own.