```java
/**
 * SimMetrics - SimMetrics is a java library of Similarity or Distance
 * Metrics, e.g. Levenshtein Distance, that provide float based similarity
 * measures between String Data. All metrics return consistant measures
 * rather than unbounded similarity scores.
 *
 * Copyright (C) 2005 Sam Chapman - Open Source Release v1.1
 *
 * Please Feel free to contact me about this library, I would appreciate
 * knowing quickly what you wish to use it for and any criticisms/comments
 * upon the SimMetric library.
 *
 * email:       s.chapman@dcs.shef.ac.uk
 * www:         http://www.dcs.shef.ac.uk/~sam/
 * www:         http://www.dcs.shef.ac.uk/~sam/stringmetrics.html
 *
 * address:     Sam Chapman,
 *              Department of Computer Science,
 *              University of Sheffield,
 *              Sheffield,
 *              S. Yorks,
 *              S1 4DP
 *              United Kingdom,
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the
 * Free Software Foundation; either version 2 of the License, or (at your
 * option) any later version.
 *
 * This program is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
 * or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
 * for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program; if not, write to the Free Software Foundation, Inc.,
 * 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 */

package uk.ac.shef.wit.simmetrics.similaritymetrics;

import uk.ac.shef.wit.simmetrics.similaritymetrics.AbstractStringMetric;

import java.io.Serializable;

/**
 * Package: uk.ac.shef.wit.simmetrics.similaritymetrics.jaro
 * Description: uk.ac.shef.wit.simmetrics.similaritymetrics.jaro implements the Jaro
String Metric.

 * Date: 01-Apr-2004
 * Time: 16:36:27
 * @author Sam Chapman <a href="http://www.dcs.shef.ac.uk/~sam/">Website</a>, <a
href="mailto:sam@dcs.shef.ac.uk">Email</a>.
 * @version 1.1
 */
public final class Jaro extends AbstractStringMetric implements Serializable {

    /**
     * a constant for calculating the estimated timing cost.
     */
    private final float ESTIMATEDTIMINGCONST = 4.12e-5f;

    /**
     * constructor - default (empty).
     */
    public Jaro() {
    }

    /**
     * returns the string identifier for the metric.
     *
     * @return the string identifier for the metric
     */
    public String getShortDescriptionString() {
        return "Jaro";
    }

    /**
     * returns the long string identifier for the metric.
     *
     * @return the long string identifier for the metric
     */
    public String getLongDescriptionString() {
        return "Implements the Jaro algorithm providing a similarity measure between
two strings allowing character transpositions to a degree";
    }

    /**
     * gets a div class xhtml similarity explaining the operation of the metric.
     *
     * @param string1 string 1
     * @param string2 string 2
     *
     * @return a div class html section detailing the metric operation.
     */
    public String getSimilarityExplained(String string1, String string2) {
        //todo this should explain the operation of a given comparison
        return null;  //To change body of implemented methods use File | Settings |
File Templates.
    }

    /**
     * gets the estimated time in milliseconds it takes to perform a similarity
timing.
     *
     * @param string1 string 1
     * @param string2 string 2
     *
     * @return the estimated time in milliseconds taken to perform the similarity
measure
     */
    public float getSimilarityTimingEstimated(final String string1, final String
string2) {
        //timed millisecond times with string lengths from 1 + 50 each increment
        //0     0.18    0.35    0.75    1.32    2.01    2.96    3.9     5.07    6.34
8.12    9.23    11.94   12.69   15.69   16.92   20.3    22.56   27.38   27.25   40.8
33.83   40.6    40.6    54.75   46.8    62.5    54.75   73      67.67   78      73
101.5   83.33   117     109.5   117.5   109     125     117.5   140.5   148.5   132.5
156.5   148.5   172     164     179.5   187.5   203     211     203     203     250
235     265     250     282     297     281
```

```
        final float str1Length = string1.length();
        final float str2Length = string2.length();
        return (str1Length * str2Length) * ESTIMATEDTIMINGCONST;
    }

    /**
     * gets the similarity of the two strings using Jaro distance.
     *
     * @param string1 the first input string
     * @param string2 the second input string
     * @return a value between 0-1 of the similarity
     */
    public float getSimilarity(final String string1, final String string2) {

        //get half the length of the string rounded up - (this is the distance used
for acceptable transpositions)
        //final int halflen = ((Math.min(string1.length(), string2.length())) / 2) +
((Math.min(string1.length(), string2.length())) % 2);
        final int halflen = (Math.max(string1.length(), string2.length())) / 2 - 1;

        //get common characters
        final StringBuffer common1 = getCommonCharacters(string1, string2, halflen);
        final StringBuffer common2 = getCommonCharacters(string2, string1, halflen);

        //check for zero in common
        if (common1.length() == 0 || common2.length() == 0) {
            return 0.0f;
        }

        //check for same length common strings returning 0.0f is not the same
        if (common1.length() != common2.length()) {
            return 0.0f;
        }

        //get the number of transpositions
        int transpositions = 0;
        for (int i = 0; i < common1.length(); i++) {
            if (common1.charAt(i) != common2.charAt(i))
                transpositions++;
        }
        transpositions /= 2.0;
        //calculate jaro metric
        return (common1.length() / ((float) string1.length()) +
                common2.length() / ((float) string2.length()) +
                (common1.length() - transpositions) / ((float) common1.length())) /
3.0f;
    }

    /**
     * gets the un-normalised similarity measure of the metric for the given strings.
     *
     * @param string1
     * @param string2
     * @return returns the score of the similarity measure (un-normalised)
     */
    public float getUnNormalisedSimilarity(String string1, String string2) {
        //todo should check this is correct (think normal metric is 0-1 scaled but
unsure)
        return getSimilarity(string1, string2);
    }
    /**
     * returns a string buffer of characters from string1 within string2 if they are
of a given
     * distance seperation from the position in string1.
     *
     * @param string1
     * @param string2
     * @param distanceSep
     * @return a string buffer of characters from string1 within string2 if they are
of a given
     *         distance seperation from the position in string1
     */
    private static StringBuffer getCommonCharacters(final String string1, final String
string2, final int distanceSep) {
        //create a return buffer of characters
        final StringBuffer returnCommons = new StringBuffer();
        //create a copy of string2 for processing
        final StringBuffer copy = new StringBuffer(string2);
        //iterate over string1
        for (int i = 0; i < string1.length(); i++) {
            final char ch = string1.charAt(i);
            //set boolean for quick loop exit if found
            boolean foundIt = false;
            //compare char with range of characters to either side
            for (int j = Math.max(0, i - distanceSep); !foundIt && j <= Math.min(i +
distanceSep, string2.length() - 1); j++) {
                //check if found
                if (copy.charAt(j) == ch) {
                    foundIt = true;
                    //append character found
                    returnCommons.append(ch);
                    //alter copied string2 for processing
                    copy.setCharAt(j, (char)0);
                }
            }
        }
        return returnCommons;
    }
}
```