```
//Project 2 - Harley Phung
//
import java.util.NoSuchElementException; //import to throw exception
public class HW2 {

    //A method that calculate the average of all values in a single array. FINISH
    public static double average (double[] array) {
        double singleAverage = 0; //find the average of the single array
        double sum = 0; //calculate the sum of the array
        if (array.length == 0) {
            throw new NoSuchElementException();
        }
        for (int index = 0; index < array.length; index = index + 1) {
            sum = sum + array[index];
        }
        singleAverage = sum / array.length;
        return singleAverage;
    }

    //  A method that calculate the average of all values in a double array.
    public static double average (double[][] array2d) {
        double doubleAverage = 0; //calculate the average of the 2-dimension array
        double sum = 0; //calculate the sum of the array
        int row; //row of the 2-dimension array
        int col; //col of the 2-dimension array
        int count = 0; //count number of index that has value
        if (array2d.length == 0) { //if there's no row
            throw new NoSuchElementException();
        }
        for(row = 0; row < array2d.length; row = row + 1){ //if there's no values
in column
            if(array2d[row].length == 0) {
                throw new NoSuchElementException();
            }
            for(col = 0; col < array2d[row].length; col = col + 1) {
                count = count + 1;
                sum = sum + array2d[row][col];
            }
        }
        doubleAverage = sum / count;
        return doubleAverage;
    }

    //A method that count number of words in the String.
    public static int countWords (String s) {
        int count = 0; //count number of words
        int index = 0;
        while (index < s.length() && s.charAt(index) == ' ') { //find whiteSpaces
            index = index + 1;
            if(index >= s.length()) {
                return 0;
            }
        }
        for ( ; index < s.length() - 1; index = index + 1) { //Count number of
words go through
            if (s.charAt(index) != ' ' && s.charAt(index + 1) == ' ') {
                count = count + 1;
            }
        }
        if (index < s.length() && s.charAt(s.length() - 1) != ' ') {
            count = count + 1;
        }
        else {
            count = count;
        }
        return count;
    }

    //A method that return a number of character of the String
    public static String truncate(String s, int l){
        StringBuilder builder = new StringBuilder();    //create a new string to
return
        //if string's length is no more than l, return it
        if (s.length() <= l){
            return s;
        }
        int conWhitespace = 0; //keep track of number of contiguous whitespace
        //string's length is more than l
        int index = 0;    //index of char that we loop over
        while (s.charAt(index) == ' '){ //if there's whitespace, do not append and
still count number of whitespace
            index = index + 1;
            conWhitespace = conWhitespace + 1;
            //if string is full of whitespace, then return empty string
            if (index == s.length()){
                //return "";
                return builder.toString();
            }
        }
        //if we get here, then s.charAt(ind) must be a character, so we find the
first char of the first word
        int temp = 0; //temporary index
        for (temp = 0; temp < conWhitespace; temp = temp + 1){
            builder.append(' ');
        }
        builder.append(s.charAt(index));
        index = index + 1;
        int moreWords = 0;
        //getting the first word no matter what, we only stop when there's no more
characters to process
        while (index < s.length() && moreWords == 0){
            //check if this word exceeds the desired length
            if (s.charAt(index) == ' '){
                //if the first word ends at l-1 or more, then we only return this
word
                if (index >= l){
                    return builder.toString();
                }
                //otherwise we need to check for more words
                else {
                    index = index + 1;
                    moreWords = 1;
                }
            }
            //if the current char is not whitespace, we simply add it to the
current first word and increment ind
            else {
                builder.append(s.charAt(index));
                index = index + 1;
            }
        }
        //ind == s.length() or more_words == 1 => end of word
        if (index == s.length()){
            return builder.toString();
        }
        StringBuilder nextWord = new StringBuilder(); //To check if the total word
exceeds the desired length
        conWhitespace = 1; //remember number of contiguous whitespace, important
later on (skipped before so have to add now)
        temp = 0;
        while (index < l){
            //if current char is whitespace, then either we have a word or it's
just whitespace before this
            if (s.charAt(index) == ' '){
                //if there's no word, simply continue
                if (nextWord.length() == 0){
                    index = index + 1;
                    conWhitespace = conWhitespace + 1;
                }
                else {
                    if (builder.length() + conWhitespace + nextWord.length() <= l){
                        //add the whitespace first
                        for (temp = 0; temp < conWhitespace; temp = temp + 1){
                            builder.append(" ");
                        }
                        //now add the new word
                        for (temp = 0; temp < nextWord.length(); temp = temp + 1){
                            builder.append(nextWord.charAt(temp));
                        }
                        //reset nextWord and the whitespace as we find a new word
                        conWhitespace = 1;
                        nextWord = new StringBuilder();
                        index = index + 1;
                    }
                    else {  //adding word will overrun limit, so return current
string instead
                        return builder.toString();
                    }
                }
            }
            else {  //else if current char is not whitespace, simply add it to
nextWord
                nextWord.append(s.charAt(index));
                index = index + 1;
            }
        }
        //we only reach here if we reach the limit. if char at index l is a
whitespace, then we can add the nextWord
        if (s.charAt(index) == ' '){
            //if there's actually a nextWord, add it in
            if (nextWord.length() > 0){
                //add the whitespace first
                for (temp = 0; temp < conWhitespace; temp = temp + 1){
                    builder.append(" ");
                }
                //now add the new word
                for (temp = 0; temp < nextWord.length(); temp = temp + 1){
                    builder.append(nextWord.charAt(temp));
                }
                //now return the string
                return builder.toString();
            }
            //otherwise just return the string
            else {
                return builder.toString();
            }
        }
        //if s.charAt(ind) is a non-whitespace char, then we disregard the current
nextWord
        return builder.toString();
    }

    // A method that return a string with evenly added whiteSpaces.
    public static String padString(String s,int l){
        StringBuilder builder = new StringBuilder();
        int countSpace = 0; //counting number of spaces
        boolean check = false; //to check if the space is in between words => false
means more than 1 space between words
        for(int index = 0; index < s.length(); index = index + 1){ // i is string
index
            if(s.charAt(index) == ' '){
                if(index == s.length() - 1){ //if the space is in the last place,
stop counting space
                    countSpace = countSpace - 1;
                }
                if(check){
                    countSpace = countSpace + 1;
                    check = false;
                }
            }
            else{ //if the (s.charAt(index) != ' ')
                check = true; // => there's one space between words
            }
        }
        if(countSpace == 0){ // less than 2 words
            return s;
        }
        check = false;
        int countSpaceBegin = 0;
        int remain = (l - s.length()) % countSpace; //remainCountSpace
        if(s.charAt(0) == ' '){
            int index = 0;
            while(index < s.length() && s.charAt(index) == ' '){// count number of
spaces before the word
                countSpaceBegin = countSpaceBegin + 1;
                index = index + 1;
            }
        }
        for(int index = s.length() - 1; index >= countSpaceBegin; index = index -
1){ //start append from end to start
            builder.append(s.charAt(index));
            if(s.charAt(index) == ' '){
                if(check){
                    if(remain > 0) {
                        for (int j = 0; j < ((l - s.length()) / countSpace) + 1; j
= j + 1) {
```

```
                            builder.append(' ');
                        }
                        remain = remain - 1;
                    }
                    else{
                        for (int j = 0; j < ((l - s.length()) / countSpace); j = j
+ 1) {
                            builder.append(' ');
                        }
                    }
                }
                check = false;
            }
            else{
                check = true;
            }
        }
        for(int index = 0; index <countSpaceBegin; index = index + 1) {
            builder.append(' ');
        }

        StringBuilder bd = new StringBuilder();
        for(int index = builder.toString().length() - 1; index >= 0; index = index
- 1) {
            bd.append(builder.toString().charAt(index));
        }

        return bd.toString();
    }

    //A method that print a String into a neat form
    public static void prettyPrint(String s , int l) {
        StringBuilder bd = new StringBuilder();
        //First we have to consider if there are whitespaces before loop.
        int index = 0;
        int numLine = 1; //check number of line printed
        int temp = 0; //remember the index of character
        //while loop that disregard whitespaces before any words
        while (index < s.length() && s.charAt(index) == ' '){
            index = index + 1;
            if (index >= s.length()){
                return;
            }
        }
        /**
         * disregard all whitespace before any word in each numLine, only start
when there's word,
         * and the pointer is at word check and append words.
         */
        while(index < s.length()) {
            //if go to here then the pointer must be at character -> append the
rest of the string here
            //If there's whitespace counted in truncate, disregard them
            while(index < s.length() && s.charAt(index) == ' ') {
                index = index + 1;
            }
            if(index == s.length()) {
                return;
            }
            temp = index;
            while(temp < s.length()) {
                bd.append(s.charAt(temp));
                temp = temp + 1;
            }
            String line = HW2.truncate(bd.toString(), l);
            // if there's only one word and it's length is more than l => append 
only that word (one possibility)
            if(line.length() > l) {
                while(index < s.length() && s.charAt(index) != ' ') { //count index 
of that word
                    index = index + 1;
                }
                System.out.println(line);
                numLine = numLine + 1;
                bd = new StringBuilder();
            }
            //if no words
            else if (line.length() <= l && HW2.countWords(line) == 0) {
                index = index + line.length();
                bd = new StringBuilder();
            }
            //if one word and length <= l
            else if(line.length() <= l && HW2.countWords(line) == 1) {
                index = index + line.length();
                System.out.println(line);
                numLine = numLine + 1;
                bd = new StringBuilder();
            }
            // if more than 2 word, length < l => padString
            else if(line.length() <= l && HW2.countWords(line) >= 2) {
                System.out.println(HW2.padString(line, l));
                index = index + line.length();
                numLine = numLine + 1;
                bd = new StringBuilder();
            }
        }
    }
}
```