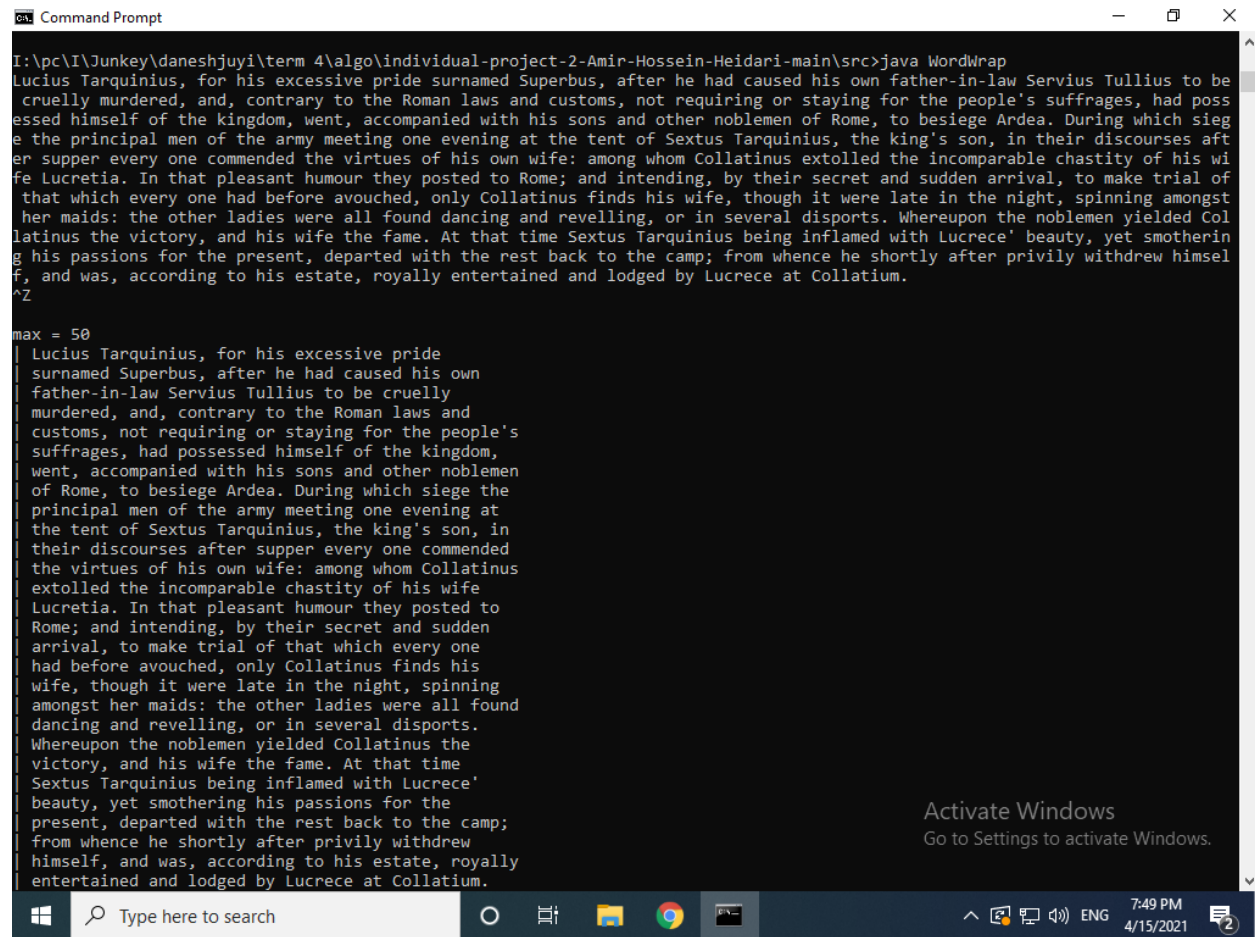## Introduction;

This is a project to wrap a given text in lines with length of 50 and making the spacing the minimum by Amir Hossein Heidari from K.N Toosi University using Java.

It consists of a class by the name of "WordWrap.

Implementation;

In the beginning user ought to enter a text and by pushing ENTER and then Ctrl z and again an ENTER, the program starts the computation and printing it out.

Here is an example given on how it works on a long poet of Shakspears.

To find the minimum of the spacing I have used dynamic programming a bottom-up approach :

$$
\text{Cost}[j] =
\begin{cases}
0 & \text{if } j=0 \\
\text{Min} (\text{Cost}[i-1] + \text{lineCost}[i][j]) & \text{if } j>0
\end{cases}
$$

In the above formula i refers to the count of words in the previous lines and j-i is the count of words in the current line

This is how it is implemented :

```java
private static int[] wordWrap (int[] input, int M){

    int n = input.length;
    int[][] extraSpace = new int[n][n];
    int[][] lineCost = new int[n][n];
    int[] cost = new int[n+1];
    int[] track = new int[n];

    for(int i=0;i<n;i++){
        extraSpace[i][i] = M - input[i];
        for(int j=i+1;j<n;j++)
            extraSpace[i][j] = extraSpace[i][j-1] - input[j] - 1;
    }

    for(int i=0;i<n;i++){
        for(int j=i;j<n;j++){
            if(extraSpace[i][j]<0)
                lineCost[i][j] = Integer.MAX_VALUE;
            else if(j==n-1)
                lineCost[i][j] = 0;
            else
                lineCost[i][j] = extraSpace[i][j]*extraSpace[i][j]*extraSpace[i][j];
        }
    }

    cost[0] = 0;
    for(int j=0;j<n;j++){
        cost[j+1] = Integer.MAX_VALUE;
        for(int i=0;i<=j;i++){
            if(cost[i]!=Integer.MAX_VALUE && lineCost[i][j]!=Integer.MAX_VALUE &&
            (cost[i] + lineCost[i][j] < cost[j+1])){
                cost[j+1] = cost[i] + lineCost[i][j];
                track[j] = i;
            }
        }
    }

    return track;
}
```

The time complexity and the memory complexity;

The time complexity is :

T(n) = 3 * n(n+1)/2 = O(n^2)

and the memory complexity is :

S(n) = n*n + n*n + n+1 + n = O(n^2)

Suggested improvements;

Changing the fixed value 50 to a final integer and get it from a user /* which is an easy job do it. :) */