19AIE105

OBJECT ORIENTED PROGRAMMING

BTECH CSE(AI) 2020-2024

AMRITA VISWA VIDHYAPEETHAM, COIMBATORE

****

DATE: - 01/03/2021

1. CB.EN. U4AIE20075 -M VISWESWARAN
2. CB.EN.U4AIE20074 – VISHNU RADHAKRISHNAN
3. CB.EN.U4AIE20072 - THUSHIT KUMAR R
4. CB.EN.U4AIE20064 - R SAIVARSHA
5. CB.EN.U4AIE20040 - MENTA SAI AKSHAY

**ACKNOWLEDGEMENT-**

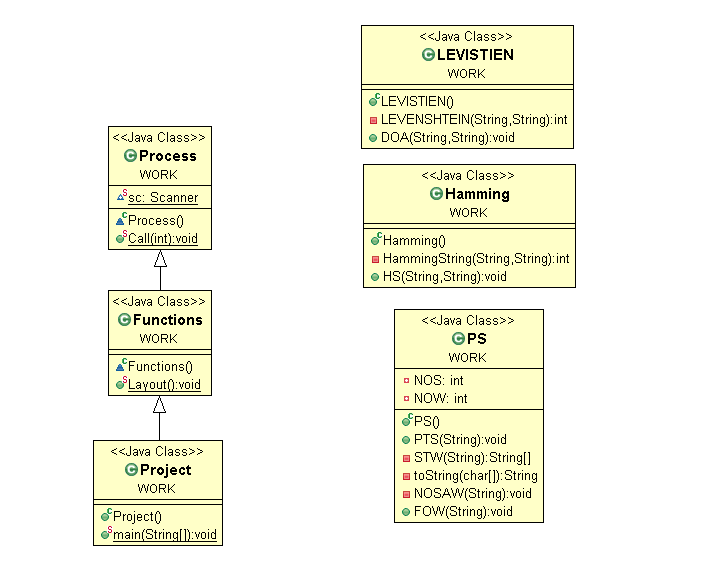
We would like to express our special thanks of gratitude to our teacher Dr. Sachin Kumar S for their able guidance and support in completing our project*.*

**TITLE:** Sentence splitter, frequency finder, hamming and levenshtein distance calculator.

**PROBLEM STATEMENT:**

1. To split the sentences of a given paragraph.
2. To find the frequency of each words occurring in the sentence.
3. To find the Hamming/Levenshtein distance when any two words are given as input depending upon the word length

**UML DIAGRAM:**

****

**THEORY:**

**Hamming Distance:**

Hamming distance is a metric for comparing two binary data strings. While comparing two binary strings of equal length, hamming distance is the number of bit positions in which the two bits are different.

The Hamming distance between two strings, a and b is denoted as d (a, b).

It is used for error detection or error correction when data is transmitted over computer networks. It is also using in coding theory for comparing equal length data words.

**Calculation of Hamming Distance**

In order to calculate the Hamming distance between two strings, and, we perform their XOR operation, (a⊕ b), and then count the total number of 1s in the resultant string.

**Example**

Suppose there are two strings 1101 1001 and 1001 1101.

11011001 ⊕ 10011101 = 01000100. Since, this contains two 1s, the Hamming distance, d (11011001, 10011101) = 2.

**Minimum Hamming Distance**

In a set of strings of equal lengths, the minimum Hamming distance is the smallest Hamming distance between all possible pairs of strings in that set.

**Example**

Suppose there are four strings 010, 011, 101 and 111.

010 ⊕ 011 = 001, d (010, 011) = 1.

010 ⊕ 101 = 111, d (010, 101) = 3.

010 ⊕ 111 = 101, d (010, 111) = 2.

011 ⊕ 101 = 110, d (011, 101) = 2.

011 ⊕ 111 = 100, d (011, 111) = 1.

101 ⊕ 111 = 010, d (011, 111) = 1.

Hence, the Minimum Hamming Distance, *dmin* = 1.

**LEVENSHTEIN DISTANCE:**

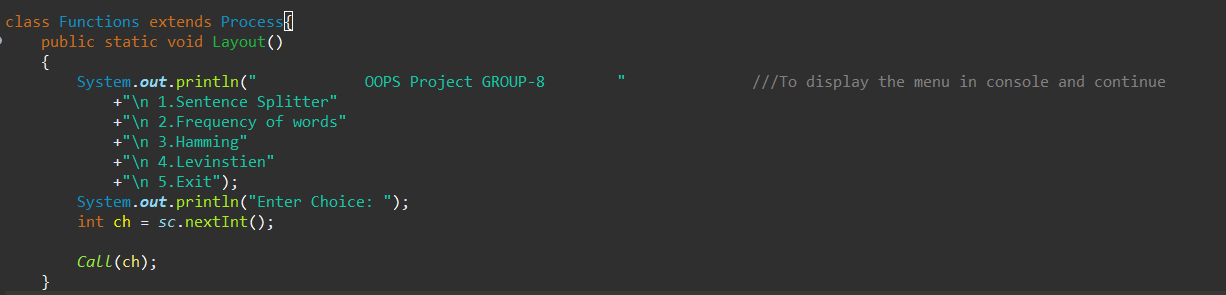
* LEVENSHTEIN DISTANCE:
* Levenshtein distance (LD) is a measure of the similarity between two strings, which we will refer to as the source string (s) and the target string (t). The distance is the number of deletions, insertions, or substitutions required to transform s into t. For example,
* If s is "test" and t is "test", then LD (s, t) = 0, because no transformations are needed. The strings are already identical.
* If s is "test" and t is "tent", then LD (s, t) = 1, because one substitution (change "s" to "n") is sufficient to transform s into t.
* The greater the Levenshtein distance, the more different the strings are.
* Levenshtein distance is named after the Russian scientist Vladimir Levenshtein, who devised the algorithm in 1965.
* The Levenshtein distance algorithm has been used in:
* Spell checking
* Speech recognition
* DNA analysis
* Plagiarism detection

**ABBREVIATION:**

* Hamming class:
  + C-hamming distance
  + String a-word 1, String b-word 2
* Levinshtien class
  + String W1, String W2-word 1, word 2 respectively
  + Char S1, char S2-character array of W1, W2 respectively
* PS class
  + NOS- number of sentences
  + NOW- number of words
* Process class
  + Sc-scanner object
  + O-PS object
  + O1-HAMMING object
  + O2-LEVISTIEN object

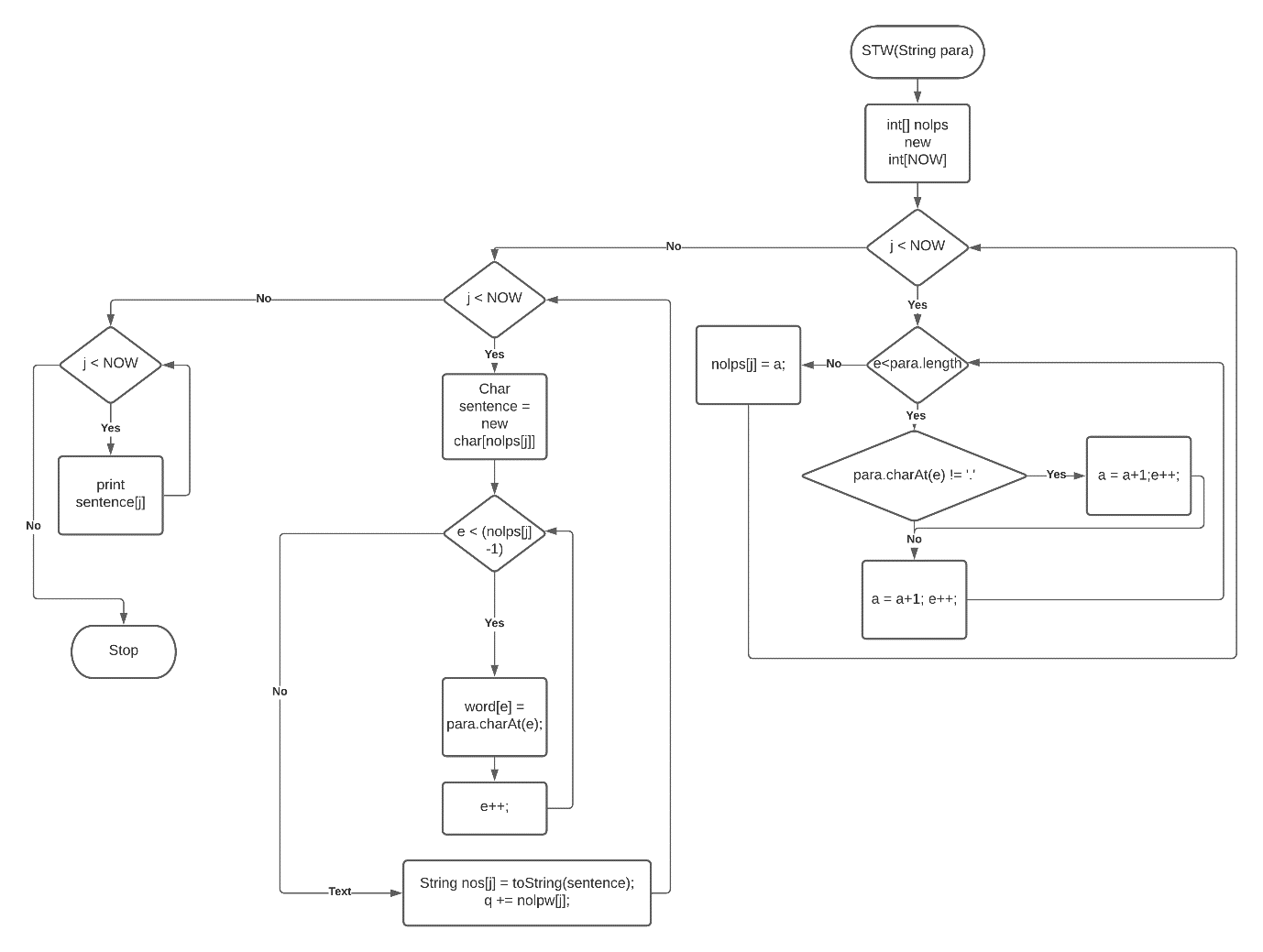
**PROPOSED APPROACH:**

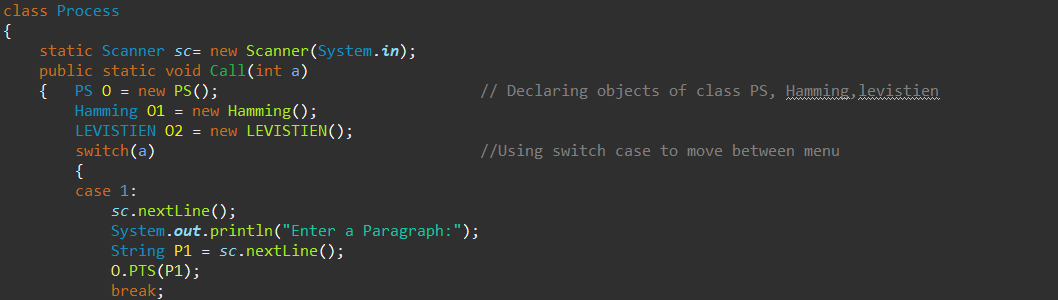
The program will go through the following steps:

1. A choice will be given to choose between sentence splitter, distance calculator, frequency finder and exit.
2. If choice =1, A paragraph will be taken and will be split into sentences using sentence splitter.
3. If choice = 2, A paragraph will be taken and frequency of each words occurring in the sentence will be displayed.
4. If choice =3, the hamming distance will be displayed for words with same length
5. If choice=4, the levenshtein distance will be displayed.
6. If choice = 5, it will be exited.

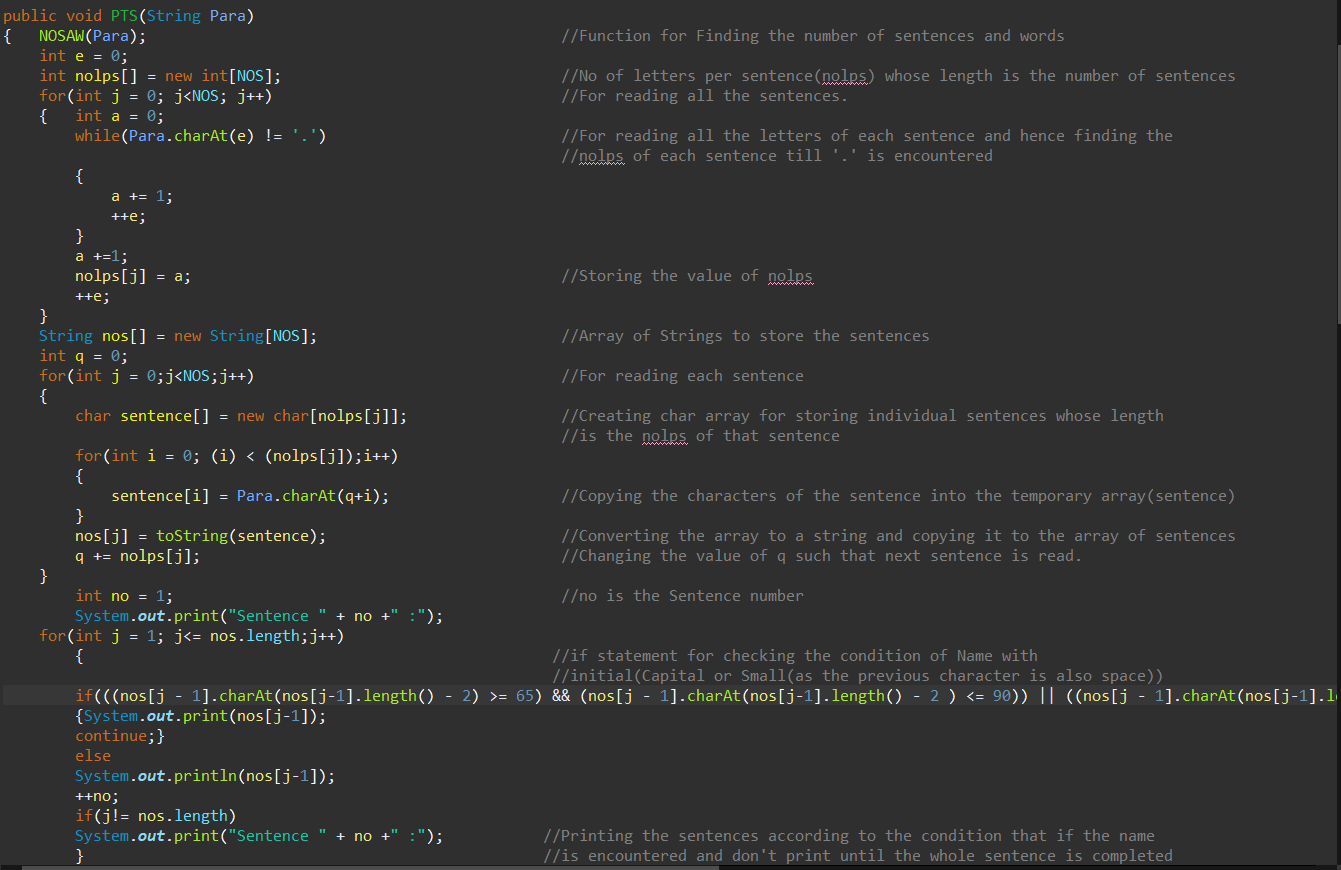
**IF CHOICE=1**

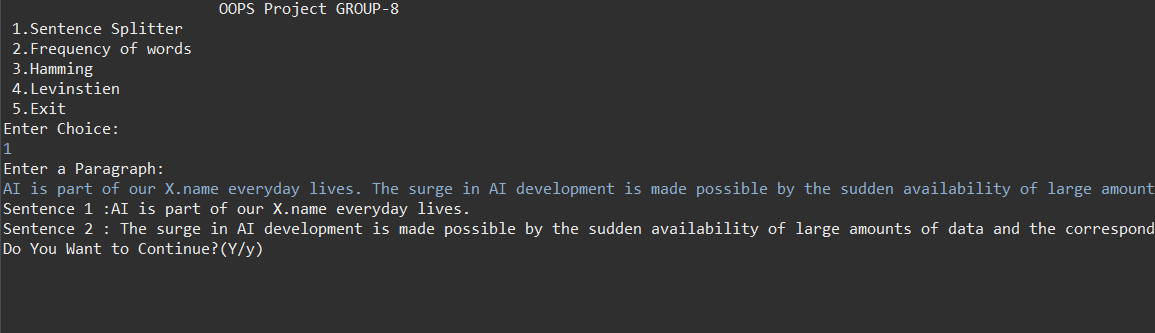
**SENTENCE SPLITTER:**

It will split the paragraphs into sentences.

An object O of class PS is created and the method PTS is called from the class PS.

* PTS is a function for Finding the number of sentences and words.
* An integer array nolps is declared to store the number of letters per sentence till a full stop (‘.’) is encountered.
* Now a string array nos is created where the sentences are stored by checking for a full stop as a determinator while ignoring the dot (‘.’) between an initial and a name.
* Now the sentences are displayed one by one.

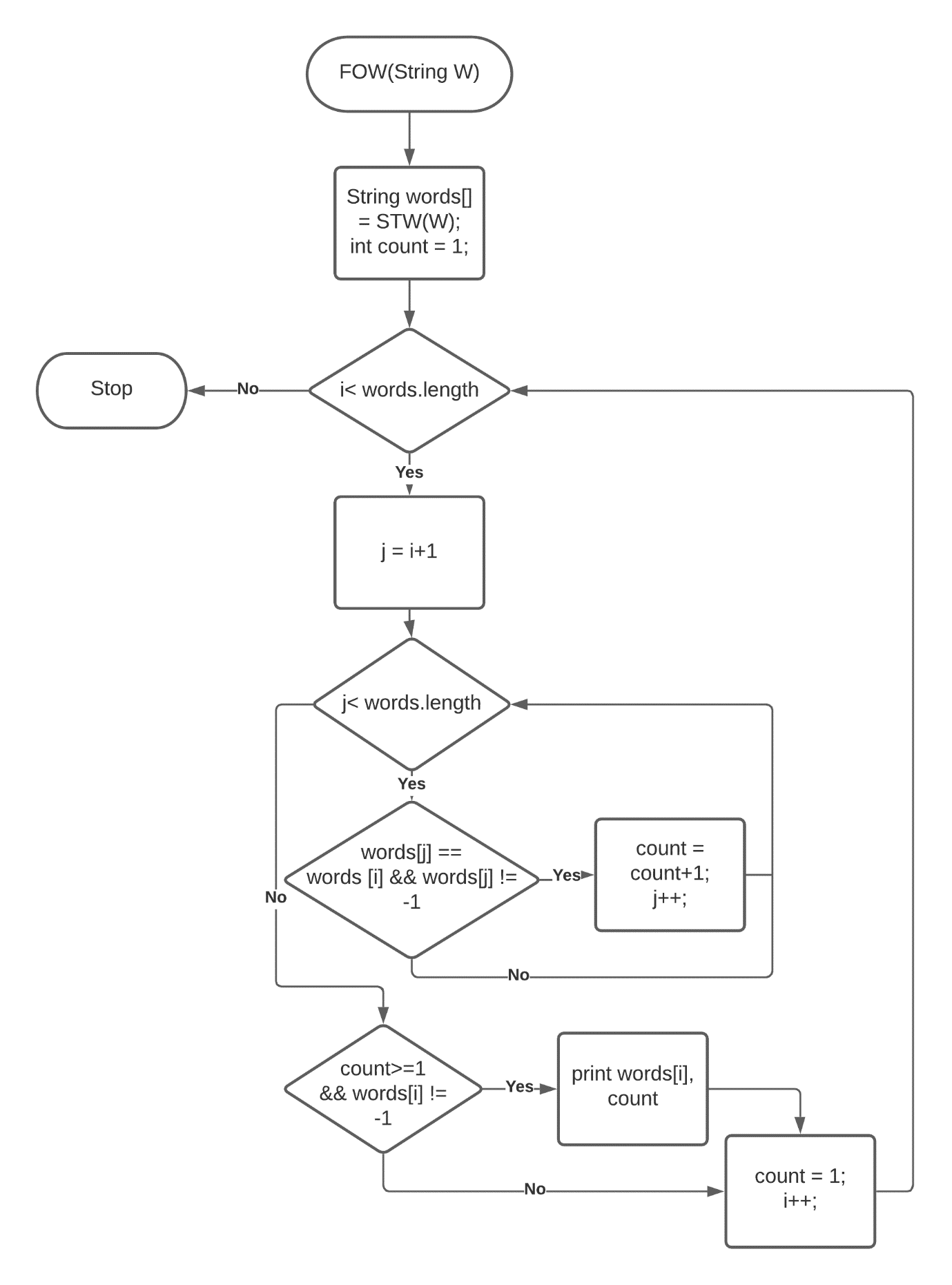
**Output**

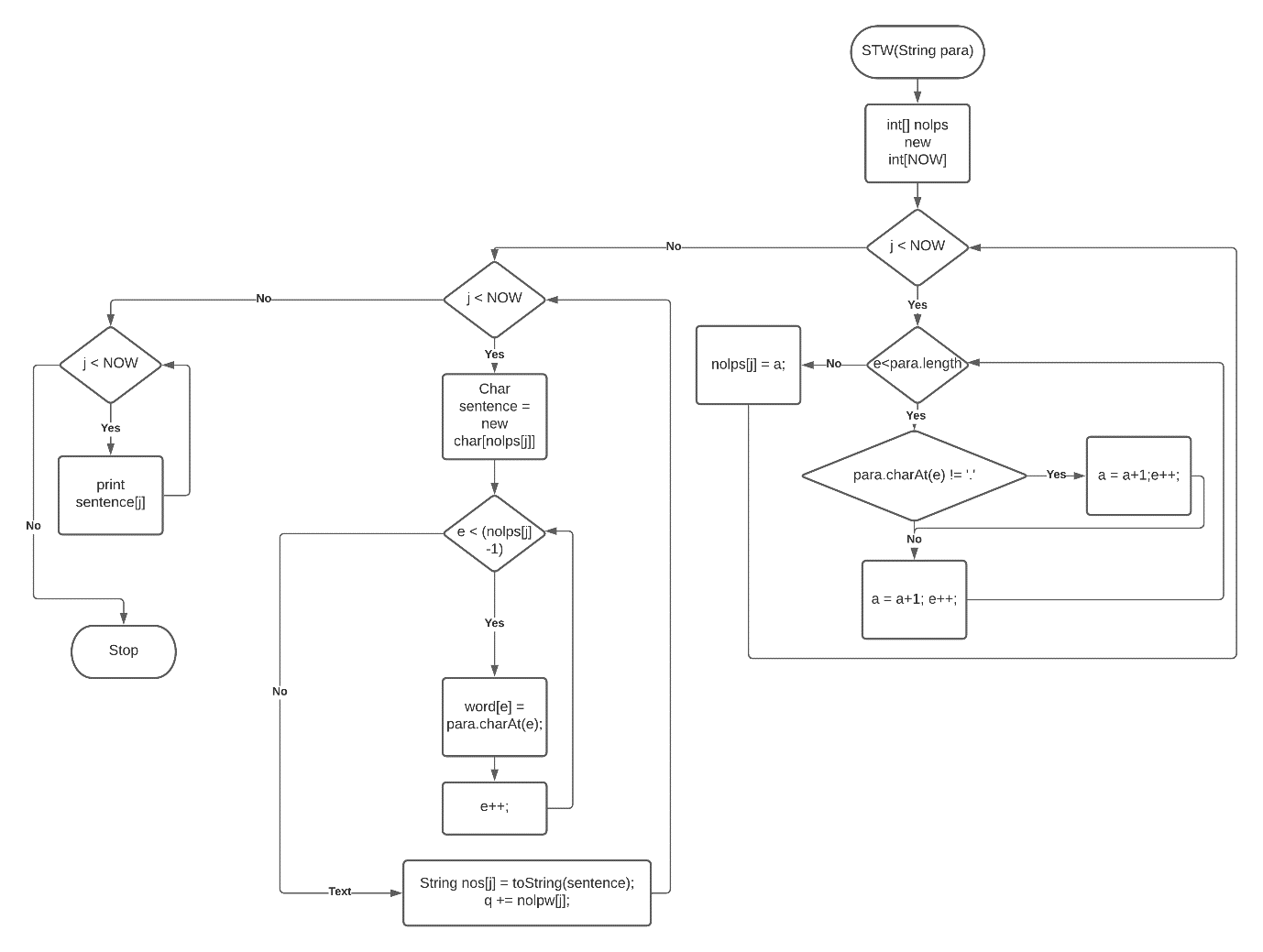
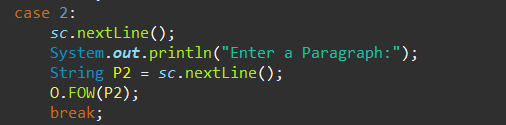
The paragraph entered: AI is part of our X.name everyday lives. The surge in AI development is made possible by the sudden availability of large amounts of data and the corresponding development and wide availability of computer systems that can process all that data faster and more accurately than humans can.

**IF CHOICE=2**

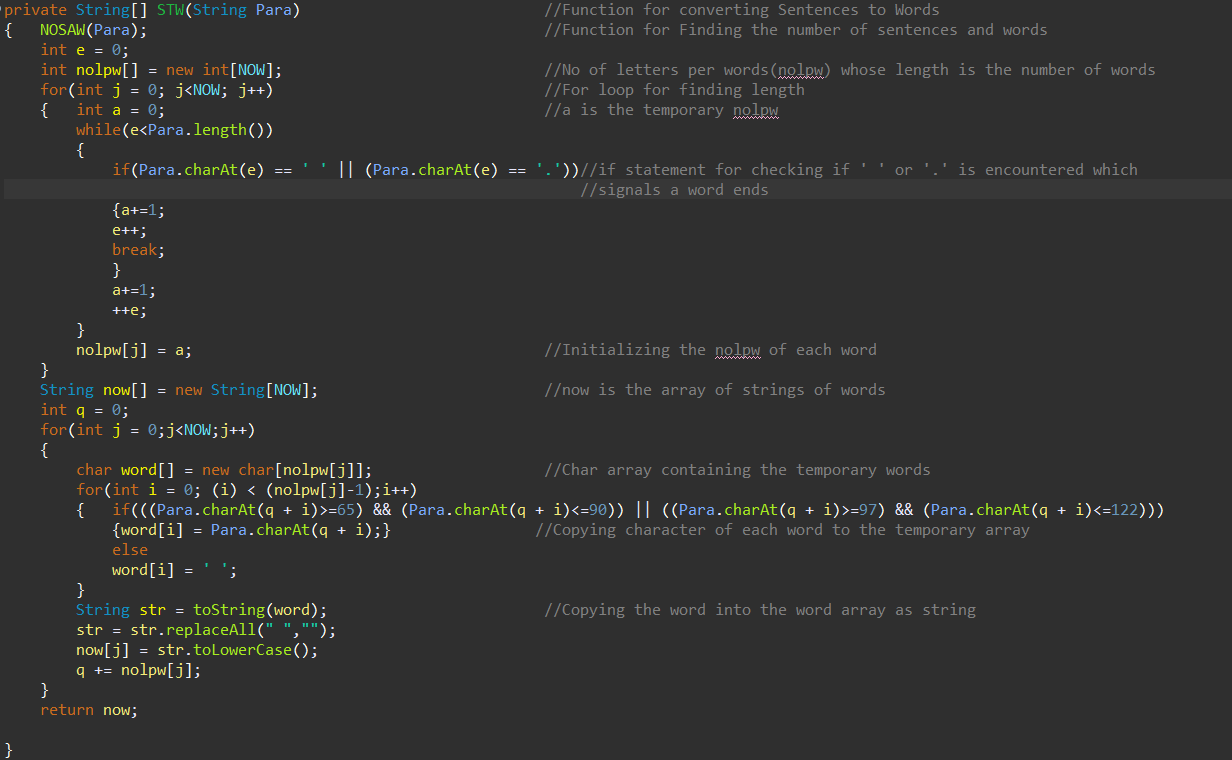
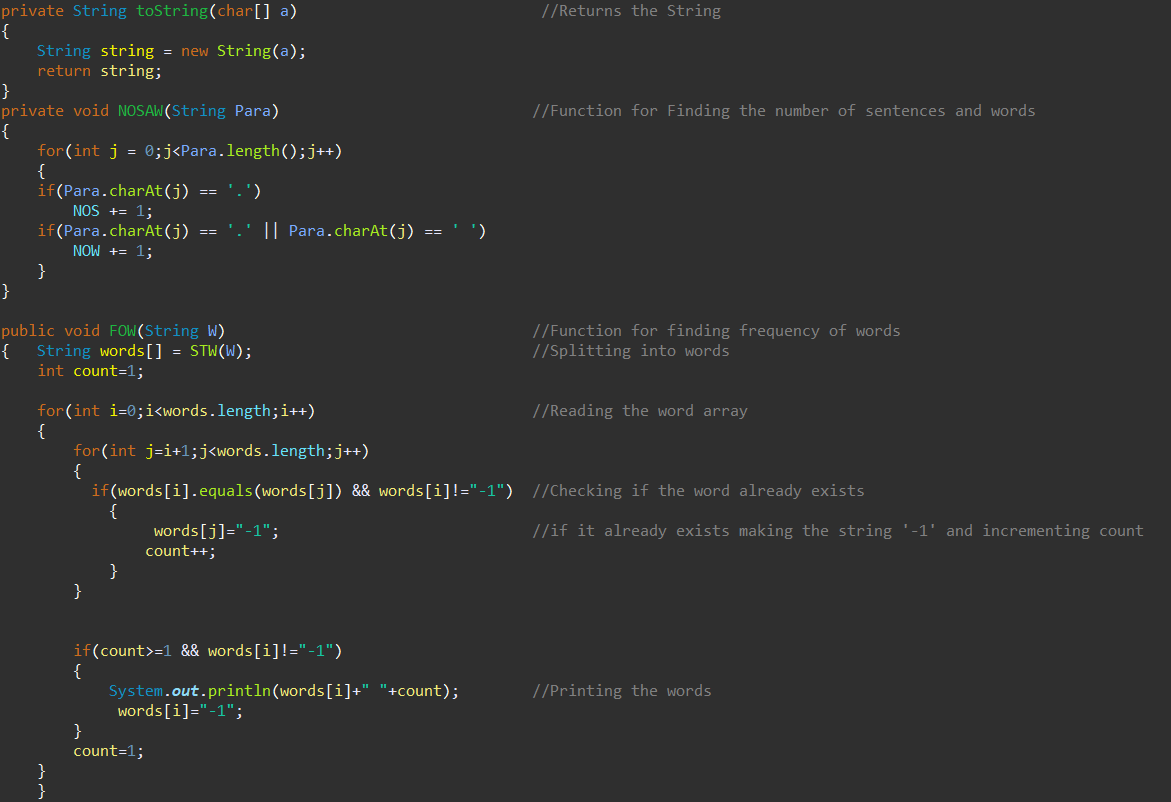
**FREQUENCY OF WORDS:**

It displays the frequency of words in a paragraph.





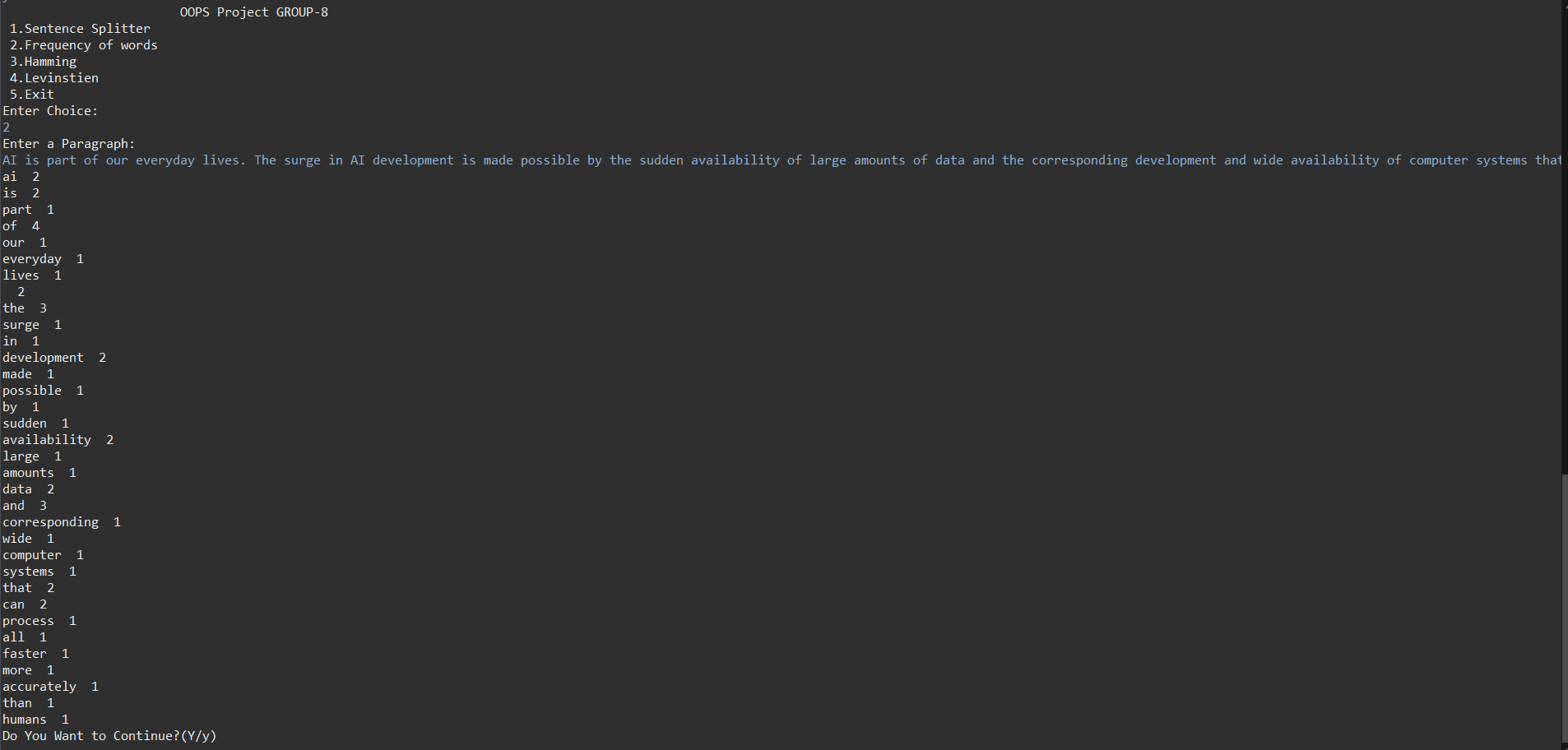
* An object O of class PS is created and the method FOW is called from the class PS.
* The sentences are split into words with the help of a method STW which converts sentences to words.
* The method STW splits the sentence into words whenever a space (‘ ‘) or a full stop (‘.’) is encountered (this doesn’t include the dot in a name).
* After getting the words we store the words in an array of strings.
* We run two for loops and check if the words are equal.
  + If they are equal, we increment the count and give -1 value for the string.
* We then display the words and count if count is greater than or equal to one and if the string is not -1.
* After displaying we change the string to -1.



**Output**

The paragraph entered

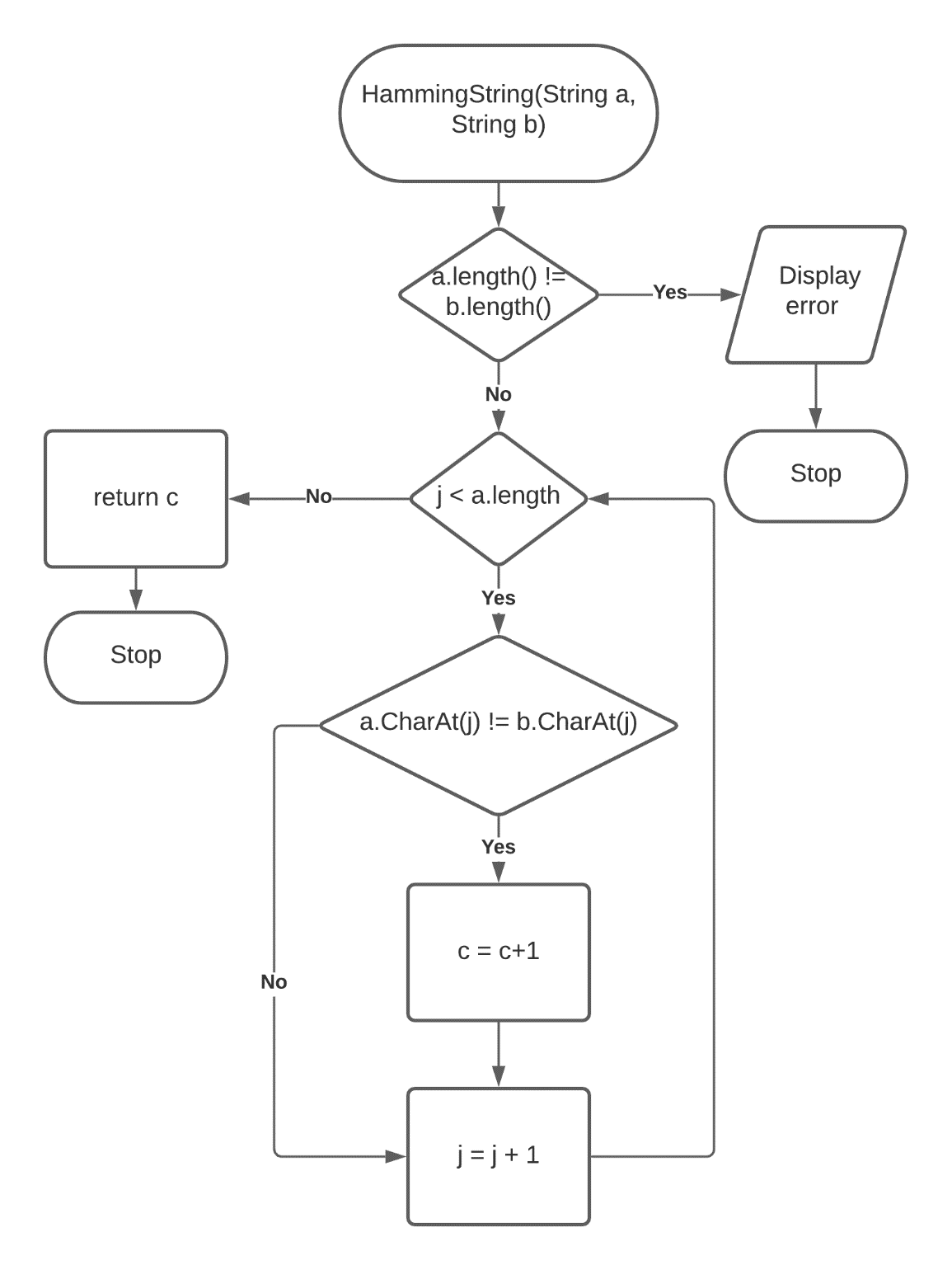
AI is part of our everyday lives. The surge in AI development is made possible by the sudden availability of large amounts of data and the corresponding development and wide availability of computer systems that can process all that data faster and more accurately than humans can.

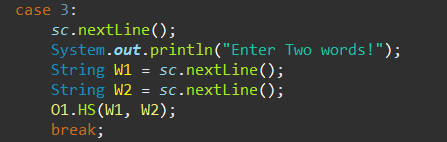


**IF CHOICE=3**

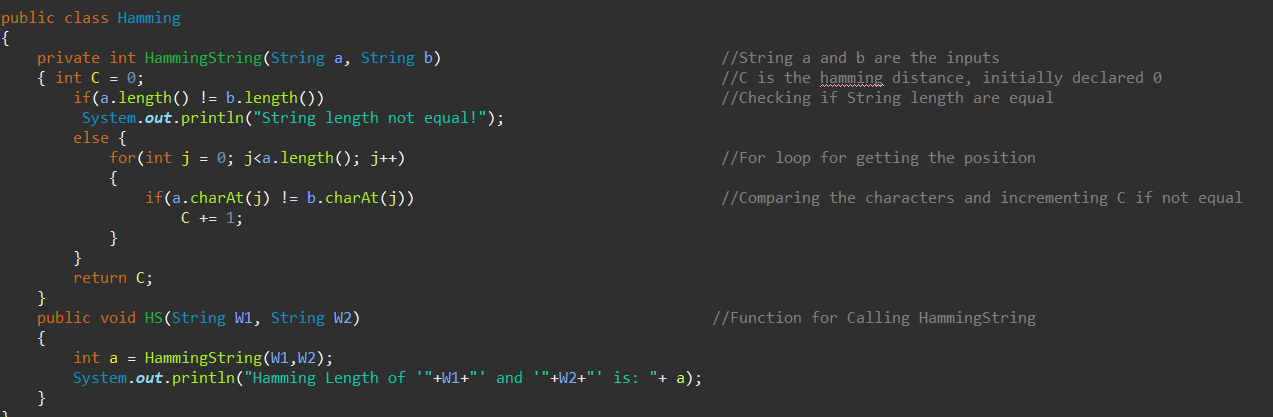
**HAMMING DISTANCE:**

The hamming distance will be displayed for words with same length.

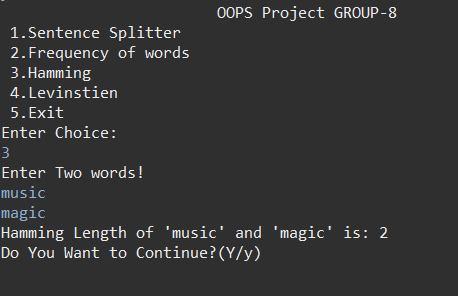




* An object O1 of class Hamming is created and the method HS is called from the class Hamming.
* The length of the strings is checked
  + If the lengths are unequal it displays “string length not equal!”.
  + Else, if the lengths are equal the strings are compared character wise and count is incremented if the characters at corresponding positions are different.
* The hamming distance is then displayed.

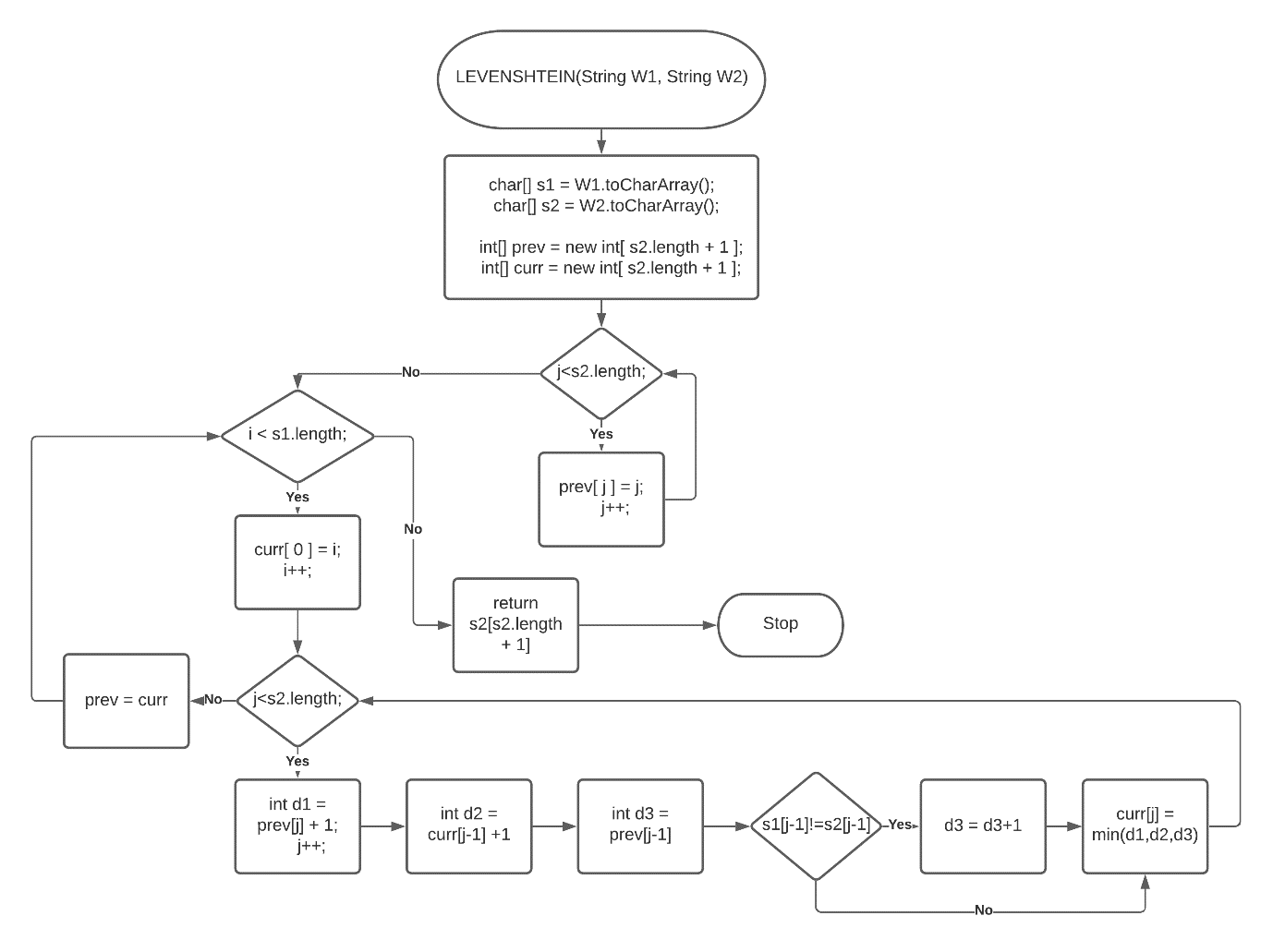


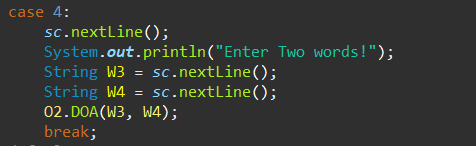
**Output**

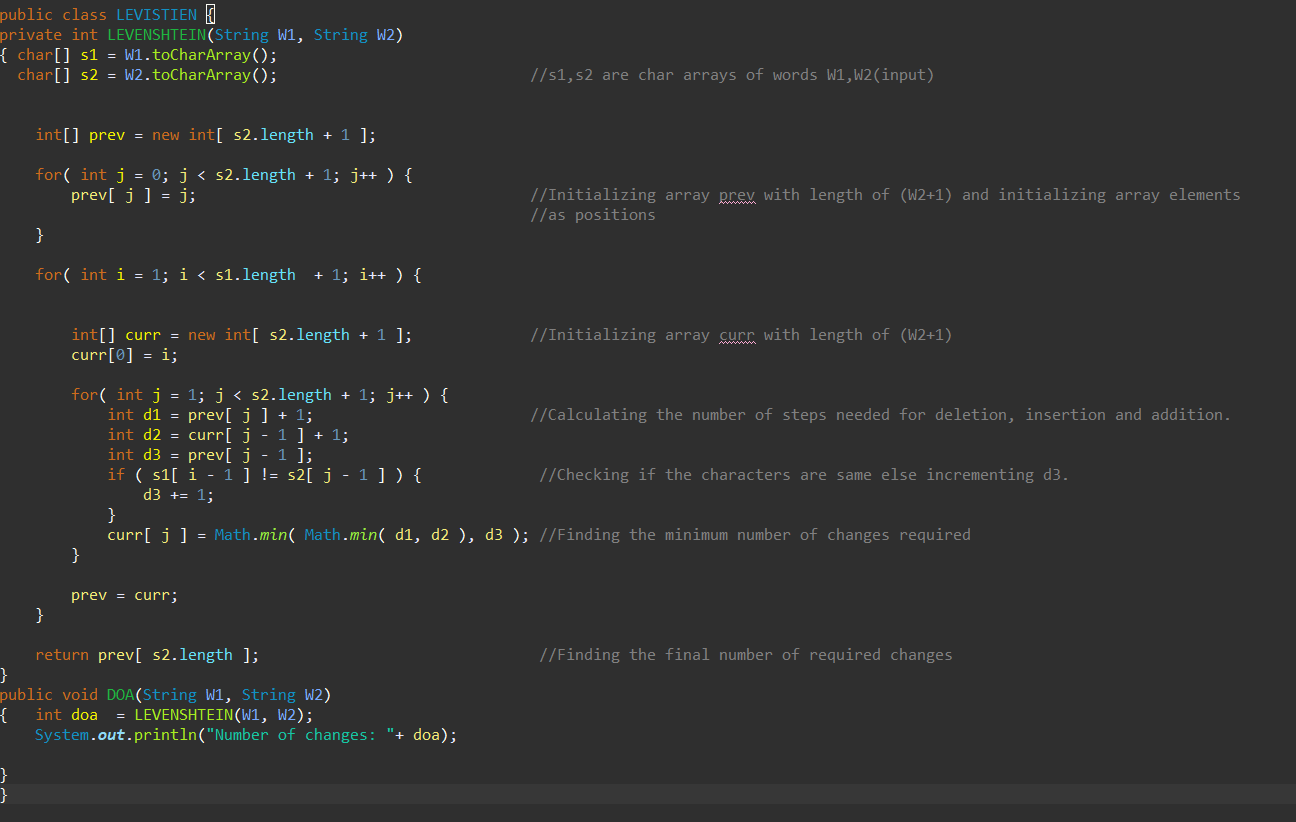
****

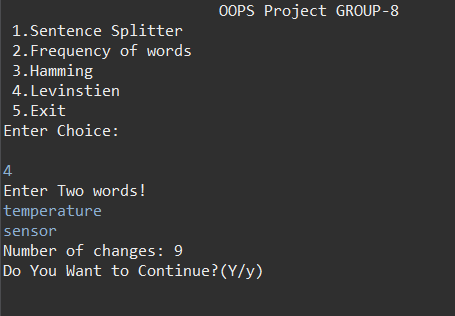
**IF CHOICE=4**

**LEVENSHTEIN DISTANCE:**

The Levenshtein distance will be displayed for words with different length.

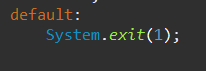
****

* An object O2 of class LEVISTIEN is created and the method DOA is called from the class LEVISTIEN.
* The two words are converted to characters.
* Two integer arrays prev and curr are declared with length of word2.
* The number of steps required for insertion, deletion and addition is calculated.
* The minimum number of changes (using Math.min command) are stored.
* ****In the method DOA the number of changes are displayed.

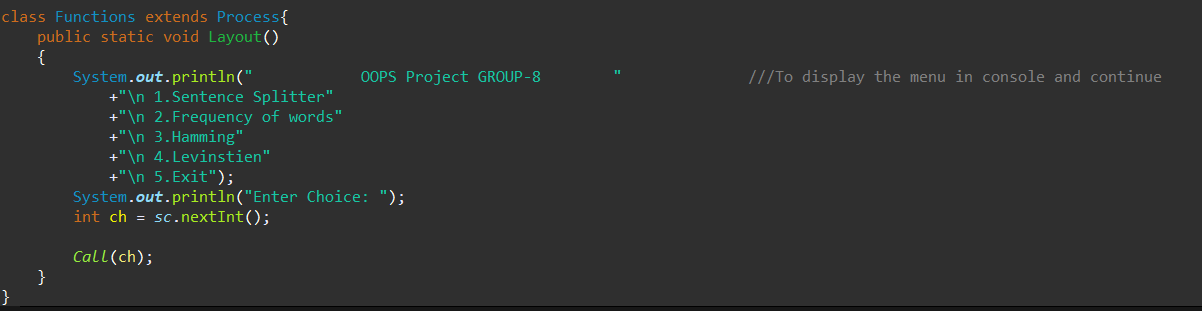
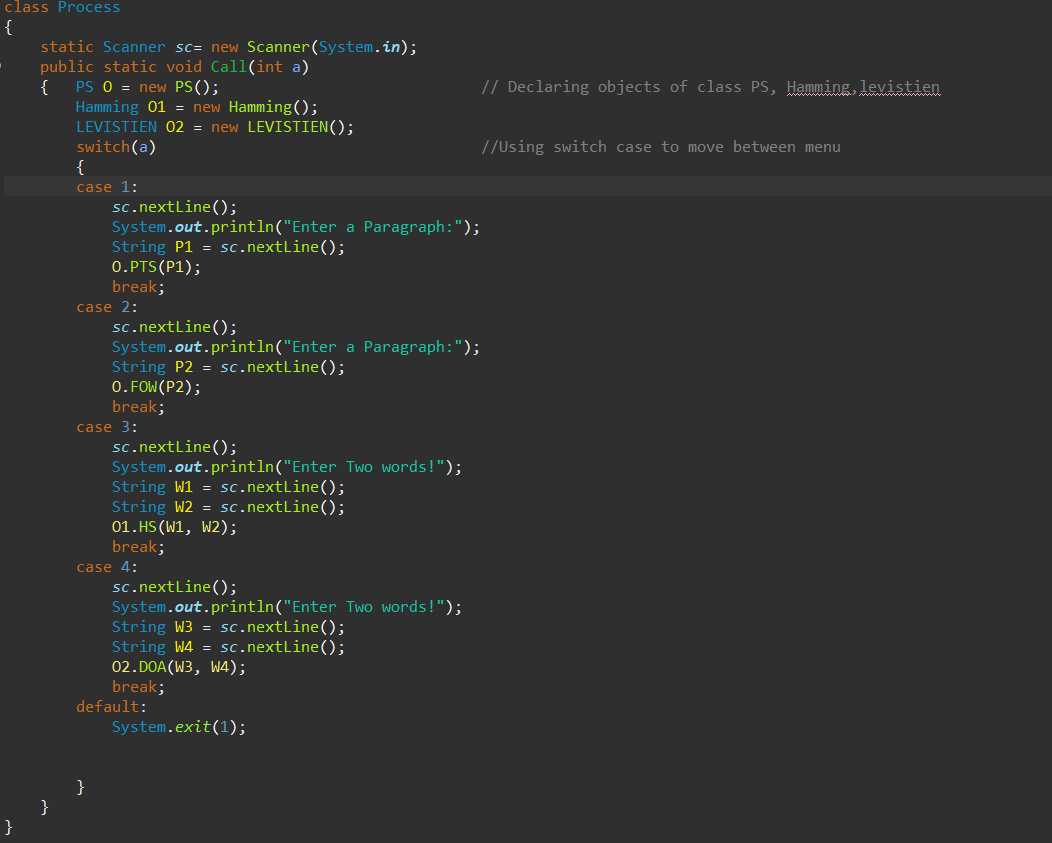
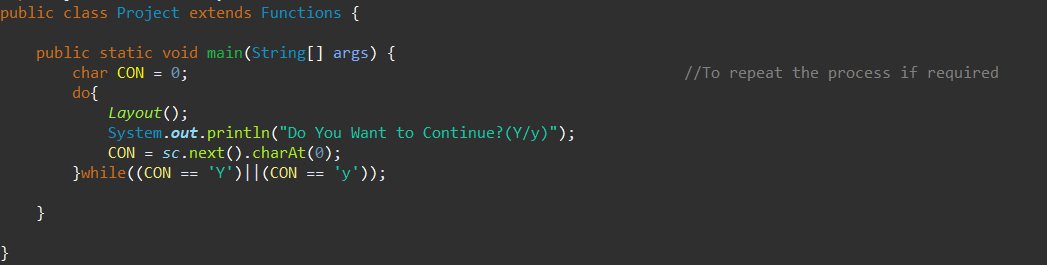
**Output**

**IF CHOICE=5**

It exits.



**THE LAYOUT OF THE PROGRAM**

* There are 3 classes in Project .java file which show multilevel inheritance.
* Class Function inherits from class process and class Project inherits from class Function.
* Class Function displays the menu.
* Class process checks the choice entered and directs the program to all other classes.
* Class Project asks if we want to continue the program or exit.

**Challenges Faced (which were later solved)**

1. The common problem was that of static allocation which had to be made dynamic with respect to the character array of input.
2. Sentence splitter
   1. The names were not correctly detected without using another if condition for checking if the previous character of ‘’.’’ was a capital letter.
3. Frequency calculator
   1. Some words were displayed again with the frequency which was later avoided after just incrementing the frequency and not adding to the array of string.
   2. The words condition had to be checked again as same condition of only ‘ ‘ and ‘.’ was not sufficient.
4. Levenshtein
   1. The simple algorithm was not enough so the algorithm had to be made to tackle the problem of calculating the distance in “changed” and “unchanged” as only two changes had to made here that is ‘u’ and ‘n’.

**CONCLUSION:** A java program with required capabilities easing manual work was created using eclipse software.

**THANK YOU**