

# WORD COUNTER



## A PROJECT REPORT

*Submitted by*

**SRINITHI D (2303811710422156)**

*in partial fulfillment of requirements for the award of the course*

**CGB1201 - JAVA PROGRAMMING**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**NOVEMBER- 2024**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY  
(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “**WORD COUNTER**” is the bonafide work of **SRINITHI D (2303811710422156)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING  
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,  
HEAD OF THE DEPARTMENT  
PROFESSOR

**SIGNATURE**

Dr.A.Delphin Carolina Rani, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

**PROFESSOR**

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram-621112.

CGB1201-JAVA PROGRAMMING  
Mr. A. MALARMANNAN A, M.E.,  
ASSISTANT PROFESSOR

**SIGNATURE**

Mr. A. Malarmannan, M.E.,

**SUPERVISOR**

**ASSISTANT PROFESSOR**

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on – 06/12/2024

CGB1201-JAVA PROGRAMMING  
Mr. M. S. SIVANANDAN, M.E.,  
INTERNAL EXAMINER  
ASSISTANT PROFESSOR

**INTERNAL EXAMINER**

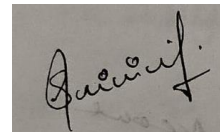
CGB1201-JAVA PROGRAMMING  
Mr. R. K. R. SIVANANDAN, M.E.,  
EXTERNAL EXAMINER  
ASSISTANT PROFESSOR  
8138-SCE, TRICHY.

**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**WORD COUNTER**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

**Signature**



---

SRINITHI D

Place: Samayapuram

Date: 06/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To serve the society by offering top-notch technical education on par with global standards

## **MISSION OF THE INSTITUTION**

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

## **VISION OF DEPARTMENT**

To be a center of eminence in creating competent software professionals with research and innovative skills.

## **MISSION OF DEPARTMENT**

**M1: Industry Specific:** To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

**M2: Research:** To prepare students for research-oriented activities.

**M3: Society:** To empower students with the required skills to solve complex technological problems of society.

## **PROGRAM EDUCATIONAL OBJECTIVES**

### **1. PEO1: Domain Knowledge**

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

### **2. PEO2: Employability Skills and Research**

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

### **3. PEO3: Ethics and Values**

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

## **PROGRAM SPECIFIC OUTCOMES (PSOs)**

### **PSO 1: Domain Knowledge**

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

### **PSO 2: Quality Software**

To apply software engineering principles and practices for developing quality software for scientific and business applications.

### **PSO 3: Innovation Ideas**

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

## **PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## ABSTRACT

This project presents a **Word Counter**, a graphical user interface (GUI) application built using Java's **Abstract Window Toolkit (AWT)**. The application allows users to input multiple lines of text, analyzes the content, and provides comprehensive results including the total number of lines, words, characters, and the frequency of each word in the input text.

The program employs robust text processing techniques, such as punctuation removal and case normalization, to ensure accurate word frequency counts. The GUI consists of an intuitive layout with a text input area, an "Analyze Text" button, and a results display section. Upon clicking the button, the program dynamically processes the input text and updates the results section.

This application serves as a practical demonstration of combining AWT-based GUI design with core Java functionalities like string manipulation, text analysis, and collections (e.g., HashMap for word frequency). It is ideal for learning AWT and can be extended for various natural language processing tasks.

**Keywords:** Java, AWT, Word Counter, Text Analysis, GUI Application



### ABSTRACT WITH POs AND PSOs MAPPING

#### CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>This project implements a <b>Word Counter</b> using Java's <b>Abstract Window Toolkit (AWT)</b>. The application analyzes user-input text to calculate total lines, words, characters, and word frequencies, addressing real-time text processing challenges. It demonstrates efficient text handling through techniques like punctuation removal, case normalization, and word frequency analysis using Java collections.</p>	<p><b>PO1 -3</b> <b>PO2 -3</b> <b>PO3 -3</b> <b>PO4 -3</b> <b>PO5 -3</b> <b>PO6 -3</b> <b>PO7 -3</b> <b>PO8 -3</b> <b>PO9 -3</b> <b>PO10 -3</b> <b>PO11-3</b> <b>PO12 -3</b></p>	<p><b>PSO1 -3</b> <b>PSO2 -3</b> <b>PSO3 -3</b></p>

Note: 1- Low, 2-Medium, 3- High

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	viii
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
<b>2</b>	<b>PROJECT METHODOLOGY</b>	
	2.1 Proposed Work	4
	2.2 Block Diagram	4
<b>3</b>	<b>MODULE DESCRIPTION</b>	
	3.1 User Interface Design	5
	3.2 Text Preprocessing	5
	3.3 Text Analysis	5
	3.4 Event Handling	5
	3.5 Output and Visualization	5
<b>4</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	
	4.1 Conclusion	6
	4.2 Future Scope	6
	<b>APPENDIX A (SOURCE CODE)</b>	7
	<b>APPENDIX B (SCREENSHOTS)</b>	10
	<b>REFERENCES</b>	11

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Objective**

The objective of this Java program is to develop a graphical user interface (GUI) application for performing advanced text analysis. Using Java's Abstract Window Toolkit (AWT), the application provides an interactive platform where users can input text, analyze its structure, and receive detailed statistics. The program calculates the total number of lines, words, and characters in the input text while also determining the frequency of each word, ignoring case and punctuation for consistency. By integrating these functionalities into a user-friendly interface with components such as text areas for input and output and a button to trigger the analysis, the application offers an efficient and intuitive tool for exploring text data. This project not only showcases the practical use of AWT for building desktop applications but also demonstrates the application of string manipulation and data structures for textual analysis.

### **1.2 Overview**

The program is a desktop application developed using Java's Abstract Window Toolkit (AWT) that allows users to perform detailed text analysis. It features a graphical user interface (GUI) where users can input text, click a button to initiate analysis, and view results in a separate output area. The application processes the input to compute the total number of lines, words, and characters, along with generating a frequency count of each word, ensuring case insensitivity and ignoring punctuation. The design focuses on simplicity and usability, demonstrating the integration of GUI components .

- A GUI for text input and analysis results.
- Functions to calculate lines, words, characters, and word frequencies.
- A simple, user-friendly interface showcasing Java's text processing and GUI capabilities.

## 1.3 Java Programming Concepts

### Basic OOP Concepts

#### Encapsulation:

- The program encapsulates functionality within classes and methods, providing a clear separation of concerns.

**Example:** The WordCounterAWT class contains all the properties (e.g., GUI components) and methods (e.g., analyzeText(), countWords()) related to the text analysis tool.

#### Abstraction:

- The program abstracts away the implementation details of text analysis by providing a simple and intuitive interface for the user.
- Users interact with the application via GUI components without needing to understand the underlying logic.

#### Inheritance:

- The AdvancedWordCounterAWT class inherits from the Frame class, leveraging Java's AWT framework to create and manage GUI components.

#### Polymorphism:

- Polymorphism is utilized in the ActionListener and WindowAdapter implementations, where methods like actionPerformed() and windowClosing() are overridden to provide custom functionality.

## Project-Specific Java Concepts

1. **Graphical User Interface (GUI):** Utilizes Java AWT components such as Frame, TextArea, Button, and Label to create an interactive and user-friendly interface.
2. **Event Handling:** Implements action listeners and window adapters to manage user interactions, such as button clicks and window events.
3. **Text Processing:** Applies string manipulation techniques, including splitting, replacing, and counting, to analyze text for word frequencies, line counts, and character counts.
4. **Data Structures:** Leverages a HashMap to efficiently store and retrieve word frequencies.
5. **Regular Expressions:** Uses regex to clean the input text by removing punctuation and ensuring case insensitivity.
6. **Modular Design:** Encapsulates functionality into methods, ensuring clarity, reusability, and maintainability of the code.

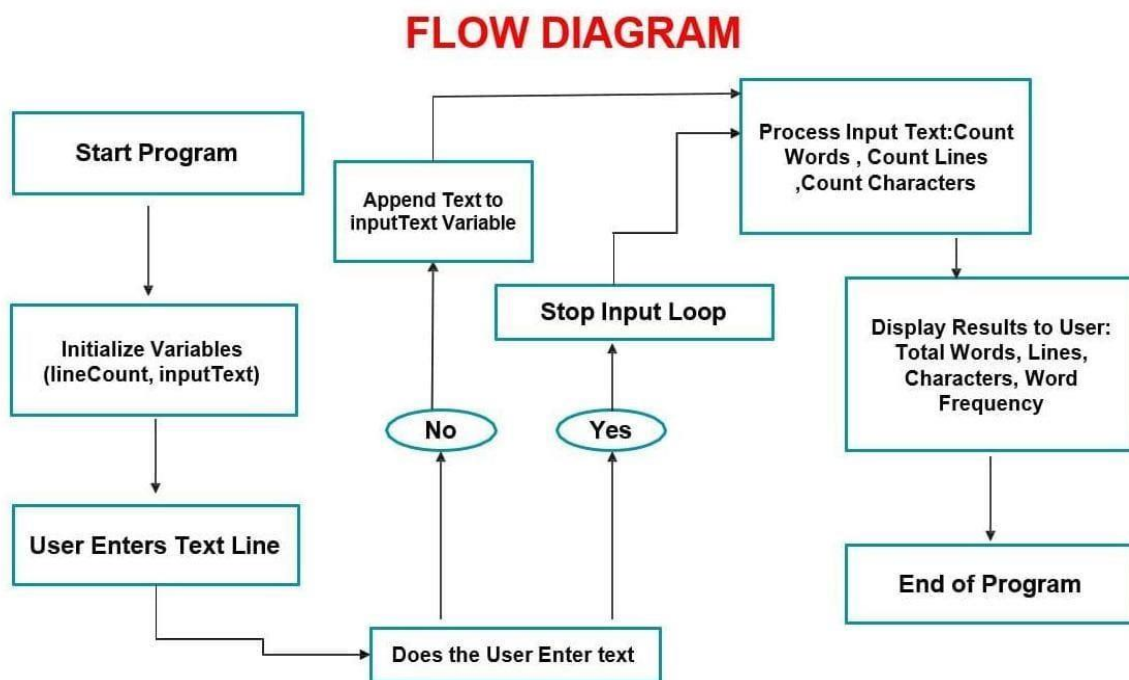
## CHAPTER 2

### PROJECT METHODOLOGY

#### 2.1 Proposed Work

The proposed work involves creating a Java-based desktop application using AWT for advanced text analysis. The application provides a user-friendly interface for inputting text and delivers statistics such as line, word, and character counts, along with word frequencies. By leveraging text processing techniques, efficient data structures, and event-driven programming, the project aims to offer an intuitive and practical tool for analyzing textual data.

#### 2.2 Block Diagram



## **CHAPTER 3**

### **MODULE DESCRIPTION**

#### **3.1 Module 1: User Interface Design**

- This module is responsible for the user interface to developed using Java's Abstract Window Toolkit (AWT), featuring a simple and intuitive design. It includes components such as a TextArea for text input, a button to trigger analysis, and an output area to display results. The layout ensures usability by organizing elements in logical sections for input, interaction, and output.

#### **3.2 Module 2: Text Preprocessing**

- Before analysis, this module handles the input text undergoes preprocessing to ensure accuracy. This includes converting text to lowercase, removing punctuation using regular expressions, and splitting it into lines or words. These steps standardize the input, making it ready for efficient analysis.

#### **3.3 Module 3: Text Analysis**

- This module manages the application performs various analyses on the processed text, including counting the total number of lines, words, and characters. Additionally, a HashMap is used to calculate the frequency of each word, providing a detailed breakdown of word usage.

#### **3.4 Module 4: Event Handling**

- This module checks for Event handling is implemented to manage user interactions. The button click event triggers the text analysis process, and a window closing event ensures proper disposal of the application. Action listeners and window adapters are used to handle these events seamlessly.

#### **3.5 Module 5: Output and Visualization**

- This module is responsible for the results of the text analysis are displayed in a read-only TextArea within the GUI. This includes statistics such as the total number of lines, words, characters, and a detailed word frequency list. The output is formatted for clarity, ensuring that users can easily interpret the results.

## CHAPTER 4

### CONCLUSION & FUTURE SCOPE

#### 4.1 CONCLUSION

The Word Counter AWT application provides a user-friendly interface for analyzing text efficiently. It integrates a variety of core functionalities such as counting lines, words, and characters, along with computing word frequencies. By leveraging Java AWT for its GUI and HashMaps for data storage, the application ensures both interactivity and robust performance. This program demonstrates key concepts in object-oriented programming, event handling, and GUI design, making it a practical and educational project. It offers a foundation for further enhancements, such as adding advanced analytics, file handling, and visualization features, making it highly extensible.

- In summary, the Word Counter AWT is a well-rounded project that combines technical accuracy with ease of use, catering to users looking for an effective and straightforward text analysis tool.

#### 4.2 FUTURE SCOPE

- **Advanced Analytics:** Add sentiment analysis, readability scores, and support for multiple languages.
- **File Handling:** Allow file uploads for analysis and export results in formats like CSV or PDF.
- **Visual Representation:** Include graphs and charts for word frequencies using JavaFX or other libraries.
- **Real-Time Features:** Provide live analysis and feedback as users type.
- **Integration:** Connect with NLP APIs and add speech-to-text capabilities.
- **Customization:** Enable user-defined parameters and personalized UI themes.



## CHAPTER 5 APPENDIX A

### (SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.util.HashMap;
import java.util.Map;

public class AdvancedWordCounterAWT extends Frame {

    // Declare GUI components
    private TextArea inputArea;
    private Button analyzeButton;
    private Label resultLabel;
    private TextArea resultArea;

    public AdvancedWordCounterAWT() {
        // Set up the Frame
        setTitle("Advanced Word Counter");
        setSize(600, 500);
        setLayout(new BorderLayout());

        // Create input area
        inputArea = new TextArea("Enter text here...", 10, 50,
TextArea.SCROLLBARS_VERTICAL_ONLY);
        add(inputArea, BorderLayout.NORTH);

        // Create analyze button
        analyzeButton = new Button("Analyze Text");
        add(analyzeButton, BorderLayout.CENTER);

        // Create result area
        Panel resultPanel = new Panel(new BorderLayout());
        resultLabel = new Label("Results:");
        resultArea = new TextArea("", 10, 50,
TextArea.SCROLLBARS_VERTICAL_ONLY);
        resultArea.setEditable(false);
        resultPanel.add(resultLabel, BorderLayout.NORTH);
        resultPanel.add(resultArea, BorderLayout.CENTER);
        add(resultPanel, BorderLayout.SOUTH);
    }
}
```

```

// Add action listener for the button
analyzeButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        analyzeText();
    }
});

// Add window listener to close the application
addWindowListener(new WindowAdapter() {
    @Override
    public void windowClosing(WindowEvent e) {
        dispose();
    }
});

// Make the frame visible
setVisible(true);
}

// Method to analyze the text
private void analyzeText() {
    String inputText = inputArea.getText();

    // Count lines
    int lineCount = inputText.split("\n").length;

    // Count words and frequencies
    Map<String, Integer> wordCounts = countWords(inputText);

    // Calculate total characters
    int charCount = inputText.length();

    // Prepare the result string
    StringBuilder result = new StringBuilder();
    result.append("Total Lines: ").append(lineCount).append("\n");
    result.append("Total Words:
").append(getTotalWordCount(wordCounts)).append("\n");
    result.append("Total Characters: ").append(charCount).append("\n\n");
    result.append("Word Frequencies:\n");
    for (Map.Entry<String, Integer> entry : wordCounts.entrySet()) {
        result.append(entry.getKey()).append(":
").append(entry.getValue()).append("\n");
    }
}

```

```

        // Display the result in the result area
        resultArea.setText(result.toString());
    }

    // Method to count words and their frequencies
    private Map<String, Integer> countWords(String inputText) {
        // Remove punctuation and convert to lowercase for case insensitivity
        inputText = inputText.replaceAll("[^a-zA-Z ]", "").toLowerCase();

        // Split the text into words using spaces as delimiters
        String[] words = inputText.split("\\s+");

        // Create a HashMap to store the word count
        Map<String, Integer> wordCounts = new HashMap<>();

        // Loop through the array of words
        for (String word : words) {
            if (word.isEmpty()) {
                continue;
            }

            // Increment the word count in the HashMap
            wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);
        }

        return wordCounts;
    }

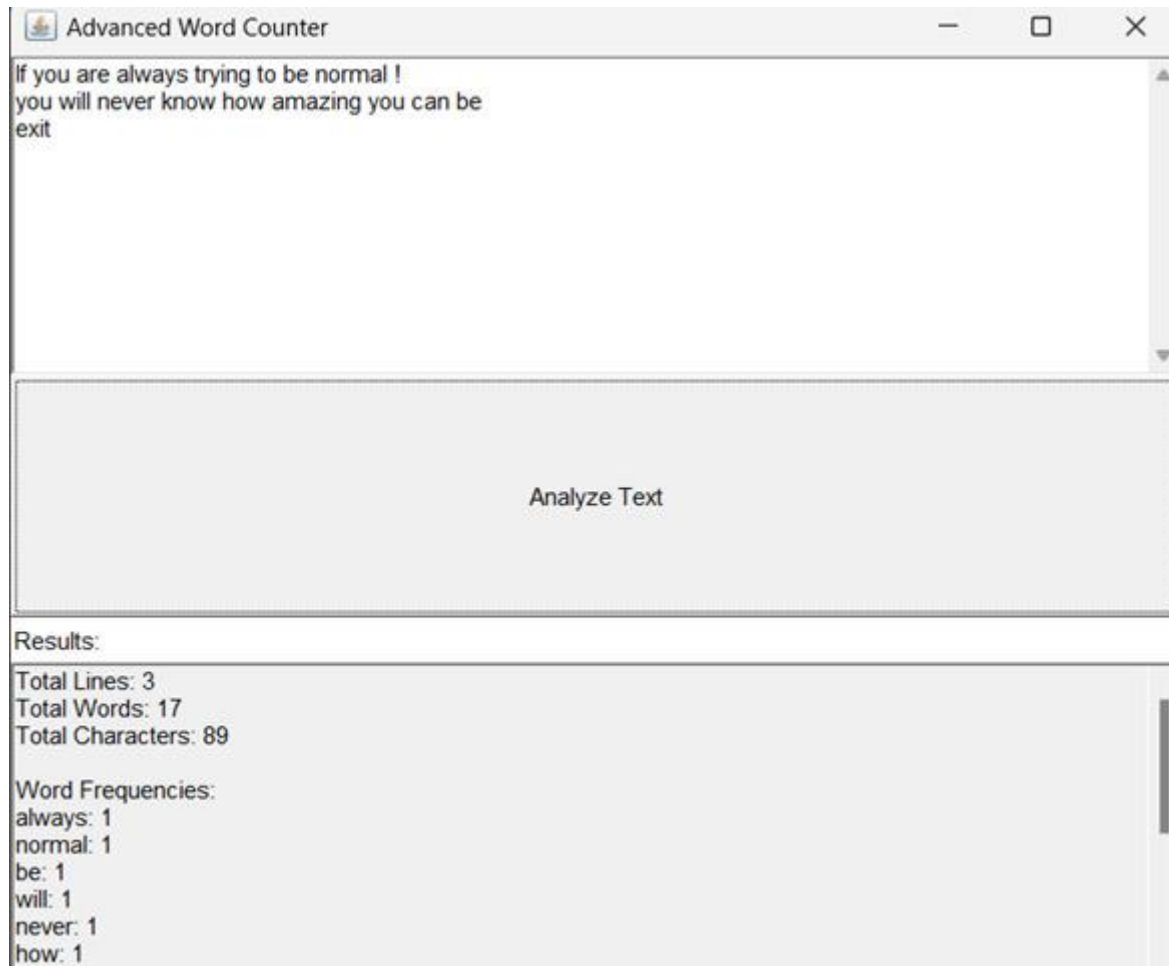
    // Method to get the total word count
    private int getTotalWordCount(Map<String, Integer> wordCounts) {
        int totalCount = 0;
        for (int count : wordCounts.values()) {
            totalCount += count;
        }
        return totalCount;
    }

    // Main method
    public static void main(String[] args) {
        new AdvancedWordCounterAWT();
    }
}

```

## APPENDIX B

### (SCREENSHOTS)



## REFERENCES

1. Java Documentation Official Java Platform Documentation by Oracle  
<https://docs.oracle.com/javase/>
2. OOP Principles and Design Patterns *Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software.*
3. Schildt, H. (2018). *Java: The Complete Reference* (11th Edition, Vol. 1, pp. 326–410). McGraw-Hill Education.