DATA STRUCTURES
CSE228
INITIAL PROJECT REPORT

Topic: A Simple Word Counter

Submitted By:
Ishaant Kumar Singh Wani
Section - K22UP
Roll no. : 47

Submitted To:
Waseem Ud Din
UID: 64339

Declaration

I hereby declare that the project work entitled ("Simple Word Counter") is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering (AI & ML) from Lovely Professional University, Phagwara, under the guidance of Waseem Ud Din Wani, during July to December 2023. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Name of Student : Ishaant Kumar Singh
Registration Number: 12203987
Project No. : 48

Signature of Student
Date: 23/10/23

Certificate

This is to certify that the declaration statement made by this student is correct to the best of my knowledge and belief. He has completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science and Engineering (AI & ML) from Lovely Professional University, Phagwara.

Signature and Name of the Mentor:
Designation:
School of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab.
Date :

Acknowledgement

I would like to extend my sincere thanks to the Lovely School of Computer Science and Engineering for providing me with the opportunity to fulfill my wish and achieve my goal. I am grateful to Waseem ud din Wani sir for providing me with the opportunity to undertake this project and for providing me with all the necessary facilities. I am highly thankful to sir for his active support, valuable time and advice, whole-hearted guidance, sincere cooperation, and pain-taking involvement during the study and in completing the assignment of preparing the said project within the time stipulated. Lastly, I am thankful to all those, particularly the various friends, who have been instrumental in creating a proper, healthy, and conducive environment and including new and fresh innovative ideas for me during the project. Without their help, it would have been extremely difficult for me to prepare the project in a timebound framework.

Introduction

Text counting is a fundamental task in many natural language processing (NLP) applications. It is used in a variety of tasks, such as text summarization, machine translation, and sentiment analysis.

In this project, we will implement a simple text counter in Java. The text counter will take a string as input and return the total number of words, characters, and lines in the string. This project is a good introduction to text processing in Java. It is also a good opportunity to learn about the basics of counting words, characters, and lines in a text. Text counting has a wide range of applications. It can be used in the following ways: To identify the most frequent words in a text, which can be useful for tasks such as text summarization and machine translation. To find the most similar texts based on their word counts, which can be useful for tasks such as document clustering and plagiarism detection. To identify the most important sentences in a text, which can be useful for tasks such as text summarization and question answering. To calculate the readability of a text, which can be useful for tasks such as educational technology and content marketing.

Problem Statement

Description : Use Java to count words, characters lines in the text.

Functionalities: Analyse text files and count words, characters, and lines

Objectives

The objective of this project is to implement a simple text counter in Java that can count the total number of words, characters, and lines in a string in an efficient and easy-to-use way. The text counter should be able to handle a wide range of text inputs, including different types of words, characters, and lines. It should also be able to handle large text inputs without sacrificing performance.

Source Code Of The Project

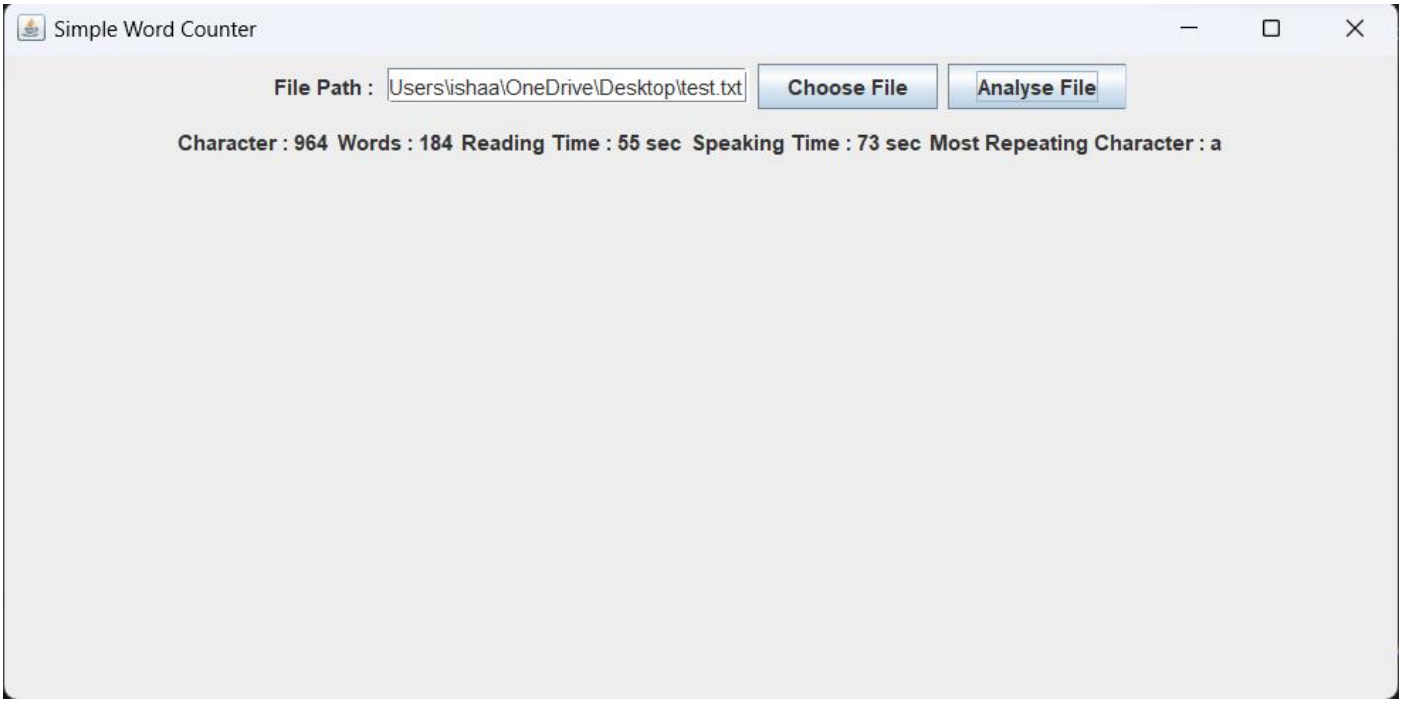
```
import java.io.File;
import java.util.*;
import java.io.IOException;
import javax.swing.*;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class MainClass {

    static StringBuilder filePath = new StringBuilder();

    public static char mostRepeatingCharacter() {
        String filePathInString = filePath.toString();
        File myFile = new File(filePathInString);
        StringBuilder s = new StringBuilder();
        try {
            Scanner in = new Scanner(myFile);
            while (in.hasNext()) {
                String word = in.next();
                s = s.append(word);
            }
            in.close();
        } catch (IOException e) {
            System.out.println("Unable to read the file");

        }

        HashMap<Character, Integer> map = new HashMap<Character, Integer>();

        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            if (map.containsKey(ch)) {
                map.put(ch, map.get(ch) + 1);
            } else {
                map.put(ch, 1);
            }
        }
        char mostFrequentChar = '\0';
        int maxFrequency = 0;

        for (Character character : map.keySet()) {
            int frequency = map.get(character);

            if (frequency > maxFrequency) {
                maxFrequency = frequency;
                mostFrequentChar = character;
            }
        }

        return mostFrequentChar;
    }

    public static int readingTime(int words) {
        int averageWPM = 200;

        double readingTimeInMin = (double) words / averageWPM;
```

```
        double readingTimeInSec = readingTimeInMin * 60;

        return (int) readingTimeInSec;
    }

    public static int speakingTime(int words) {
        int averageWPM = 150;

        double speakingTimeInMin = (double) words / averageWPM;

        double speakingTimeInSec = speakingTimeInMin * 60;

        return (int) speakingTimeInSec;
    }

    public static int characterCounter() {
        String filePathInString = filePath.toString();
        File myFile = new File(filePathInString);
        int count = 0;
        try {
            Scanner in = new Scanner(myFile);
            while (in.hasNext()) {
                String word = in.next();
                count = count + word.length();
            }
            in.close();
        } catch (IOException e) {
            System.out.println("Unable to read the file");

        }
        return count;
    }

    public static int wordCounter() {
        String filePathInString = filePath.toString();
        File myFile = new File(filePathInString);
        int count = 0;
        try {
            Scanner in = new Scanner(myFile);
            while (in.hasNext()) {
                in.next();
                count++;
            }
            in.close();
        } catch (IOException e) {
            System.out.println("Unable to read the file");

        }
        return count;
    }

    public static void main(String[] args) {
        // StringBuilder filePath = new StringBuilder();
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                JFrame f = new JFrame("Simple Word Counter");
                f.setVisible(true);
                f.setSize(800, 400);
                f.setLayout(new FlowLayout());
                JLabel l1 = new JLabel("File Path : ");
                f.add(l1);
                JTextField t1 = new JTextField(20);
```

```
                f.add(t1);
                JButton b1 = new JButton("Choose File");
                f.add(b1);
                b1.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        // StringBuilder filePath = new StringBuilder();
                        JFileChooser chooser = new JFileChooser();
                        int result = chooser.showOpenDialog(f);
                        if (result == JFileChooser.APPROVE_OPTION) {
                            t1.setText(chooser.getSelectedFile().getAbsolutePath());
                            filePath = filePath.append(chooser.getSelectedFile().getAbsolutePath());
                        }
                    }
                });
                JButton b2 = new JButton("Analyse File");
                f.add(b2);
```

```
                b2.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        JPanel responsePanel = new JPanel(new FlowLayout());
                        int characterInTheFile = characterCounter();
                        int wordsInTheFile = wordCounter();
                        int readingTime = readingTime(wordsInTheFile);
                        int speakingTime = speakingTime(wordsInTheFile);
                        char mostFrequentChar = mostRepeatingCharacter();
                        responsePanel.add(new JLabel("Character : " + characterInTheFile));
                        responsePanel.add(new JLabel("Words : " + wordsInTheFile));
                        responsePanel.add(new JLabel("Reading Time : " + readingTime + " sec"));
                        responsePanel.add(new JLabel("Speaking Time : " + speakingTime + " sec"));
                        responsePanel.add(new JLabel("Most Repeating Character : " + mostFrequentChar));
                        f.add(responsePanel);
                        f.revalidate();
                        f.repaint();
                    }
                });
                f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            }
        });
    }
}
```

Output Of The Project



GitHub Link Of The Project

GitHub Link - https://github.com/ishaant-97/wordCounter-DSA-Project

Scope Of The Project

The scope of this project is to implement a simple text counter in Java that can count the total number of words, characters, and lines in a string. The text counter should be efficient and easy to use, and it should be able to handle a wide range of text inputs, including different types of words, characters, and lines. The text counter should also be able to handle large text inputs without sacrificing performance. The text counter should be easy to use. The text counter should also be efficient. The text counter should be able to count the words, characters, and lines in a text quickly and efficiently, even for large text inputs.

Conclustion

In this project, we implemented a simple text counter in Java that can count the total number of words, characters, and lines in a string. The text counter is efficient and easy to use, and it can handle a wide range of text inputs, including different types of words, characters, and lines. The text counter can also handle large text inputs without sacrificing performance. The text counter can be used for a variety of tasks, such as identifying the most frequent words in a text, finding the most similar texts based on their word counts, identifying the most important sentences in a text, and calculating the readability of a text. The text counter can also be used in a variety of other applications, such as natural language processing (NLP) tasks, educational technology tools, content marketing tools, and plagiarism detection tools. We believe that the text counter is a valuable tool that can be used for a variety of purposes. We hope that the text counter will be used by others to improve their understanding of text data and to develop new and innovative applications.

THANK YOU