**A picture containing shape

Description automatically generated**

**a**

**MINI PROJECT REPORT ON**

**“Text Editor for java”**

By

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **NAME** | **ROLL NO.** |
| **1)** | **Alok Agrawal** | **32403** |
| **2)** | **Krishna Chidrawar** | **32409** |
| **3)** | **Mohit Dulani** | **32420** |
| **4)** | **Chinmay Gate** | **32423** |

**GUIDE**

**Mrs. Bhakti Kadam**

**Course: Fundamentals of JAVA Programming**

**Department Of**

**Electronics and TELECommunication ENGINEERING**

**PUNE INSTITUTE OF COMPUTER TECHNOLOGY**

**PUNE – 43**

**A. Y. 2022-23**

**K**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| **1** | **Problem Statement** | **1** |
| **2** | **Objectives** |  |
| **3** | **Introduction** |  |
| **4** | **Source code with Flowchart** |  |
| **5** | **Result** |  |
| **6** | **Conclusion** |  |
| **7** | **Applications** |  |
| **8** | **Future scope** |  |
| **9** | **Copy Right Affirmation** |  |

**1. PROBLEM STATEMENT:**

In day-to-day life we come across the tasks where we are supposed to do lots of typing work in that case the user need to spend lots of time in typing. So, we thought to build a generalize text editor which will suggest the words according to the user input. The word text editor program is similar to a cell phone or email word suggester. User may enter a sub-word, press ENTER, and if the sub-word matches with already available words in dictionary the program will suggest the user to select his choice.

**2. OBJECTIVE:**

1. The objectives of the project were to build a text editor where user can note, type fluently. Basically the objective was to build a notepad where we will provide option to save, edit, cut, copy and paste the text . We will be using ActionListener for this purpose.
2. The second task is to get suggestions from the given dictionary and user will be able to select the desired word he wants. The Autocompletion algorithm will suggest used to suggest the words and complete the text by merging the two algorithms which will give us the final output.

**3. INTRODUCTION**:

* 1. **Background/context**

The text editor is a type of program used for editing pain text files. A plain text file is represented and edited by showing all the characters as they are present in the file . The most commonly used character set is ASCII , especially recently , as plain text files are more often being used for programming and configurations , and less frequently for documentation.

* 1. Relevance

Relevance of the project/topic chosen to the field of Electronics and Communication engineering and related subjects based on the curriculum is to be clearly explained here

* 1. Project Details :

This editor is a simple editor, very similar to that of a notepad editor that extends the basic features to the end-user like:

* File open - User can either open the files already existing n the system or open a new blank file
* Files save - one can save the file in any desired format like- txt, doc, java etc. The file is stored in the location specified by the user.
* Save As - One can save the file in any desired format like-txt, doc, java etc. The file is stored in any other location specified by the user.
* Cut-Copy-Paste - This editor also lets the user cut-copy-paste the edited text.
* Undo - The user is allowed to undo the text edited. This feature allows letter by letter undo.
* Redo The user is allowed to redo the text edited This feature allows letter by letter Redo.
* Print – It prints the content of the file or saves them in txt format.
  + 1. OVERVIEW:

This project "Text Editor" is software which can edit plain text. It is made using Java Swings and AWT. In this project all the frames are designed in Swing. Today most programmers use Swing. Swing is a set of classes that provides more powerful and flexible GUI components than does the AWT Swing provides the look and feel of the modern Java GUI .

Swing did not exist in the early days of Java. Rather, it was a response to deficiencies present in Java's original GUI subsystem, the Abstract Window Toolkit. The AWT defines a basic set of controls, windows, and dialog boxes that support a usable, but limited graphical interface

*SWING-OVERVIEW :*

Swing API is a set of extensible GUI Components to case the developer's life to create JAVA based Front End/GUI Applications. It is build on top of AWT API and acts as a replacement of AWT API, since it has almost every control corresponding to AWT controls.

Swing component follows a Model-View-Controller architecture to fulfill the following criteria’s.

* A single API is to be sufficient to support multiple look and feel .
* API is to be model driven so that the highest-level API is not required to have data .

3.4 Scope:

**4.SOURCE CODE:**

Paste working source code and illustrate the same

// Java Program to create a text editor for java using autocorrection of words

import javax.swing.\*;

import java.io.\*;

import java.awt.event.\*;

import javax.swing.plaf.metal.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

//This is action listner class which will implement all the menu items and its functiions

class editor extends JFrame implements ActionListener {

//For texting purpose

JTextArea t;

//Java frame

JFrame f;

// Constructor

editor()

{

// Create a frame

f = new JFrame("Notepad");

try {

// Set metal look and feel

UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

// Set theme to ocean

MetalLookAndFeel.setCurrentTheme(new OceanTheme());

}

catch (Exception e) {

}

t = new JTextArea();

// Create a menubar

JMenuBar mb = new JMenuBar();

// Create sub-menu for menu

JMenu m1 = new JMenu("File");

// Create menu items

JMenuItem mi1 = new JMenuItem("New");

JMenuItem mi2 = new JMenuItem("Open");

JMenuItem mi3 = new JMenuItem("Save");

JMenuItem mi9 = new JMenuItem("Print");

// Add action listener

mi1.addActionListener(this);

mi2.addActionListener(this);

mi3.addActionListener(this);

mi9.addActionListener(this);

m1.add(mi1);

m1.add(mi2);

m1.add(mi3);

m1.add(mi9);

// Create amenu for menu

JMenu m2 = new JMenu("Edit");

// Create menu items

JMenuItem mi4 = new JMenuItem("cut");

JMenuItem mi5 = new JMenuItem("copy");

JMenuItem mi6 = new JMenuItem("paste");

mi4.addActionListener(this);

mi5.addActionListener(this);

mi6.addActionListener(this);

m2.add(mi4);

m2.add(mi5);

m2.add(mi6);

JMenuItem mc = new JMenuItem("close");

mc.addActionListener(this);

mb.add(m1);

mb.add(m2);

mb.add(mc);

f.setJMenuBar(mb);

f.add(t);

f.setSize(500, 500);

f.show();

}

public void actionPerformed(ActionEvent e)

{

String s = e.getActionCommand();

if (s.equals("cut")) {

t.cut();

}

else if (s.equals("copy")) {

t.copy();

}

else if (s.equals("paste")) {

t.paste();

}

else if (s.equals("Save")) {

JFileChooser j = new JFileChooser("f:");

int r = j.showSaveDialog(null);

if (r == JFileChooser.APPROVE\_OPTION) {

File fi = new File(j.getSelectedFile().getAbsolutePath());

try {

FileWriter wr = new FileWriter(fi, false);

BufferedWriter w = new BufferedWriter(wr);

w.write(t.getText());

w.flush();

w.close();

}

catch (Exception evt) {

JOptionPane.showMessageDialog(f, evt.getMessage());

}

}

else

JOptionPane.showMessageDialog(f, "the user cancelled the operation");

}

else if (s.equals("Print")) {

try

{

t.print();

}

catch (Exception evt) {

JOptionPane.showMessageDialog(f, evt.getMessage());

}

}

else if (s.equals("Open")) {

JFileChooser j = new JFileChooser("f:");

int r = j.showOpenDialog(null);

if (r == JFileChooser.APPROVE\_OPTION) {

File fi = new File(j.getSelectedFile().getAbsolutePath());

try {

// String

String s1 = "", sl = "";

// File reader

FileReader fr = new FileReader(fi);

try (// Buffered reader

BufferedReader br = new BufferedReader(fr)) {

// Initialize sl

sl = br.readLine();

// Take the input from the file

while ((s1 = br.readLine()) != null) {

sl = sl + "\n" + s1;

}

}

// Set the text

t.setText(sl);

}

catch (Exception evt) {

JOptionPane.showMessageDialog(f, evt.getMessage());

}

}

// If the user cancelled the operation

else

JOptionPane.showMessageDialog(f, "the user cancelled the operation");

}

else if (s.equals("New")) {

t.setText("");

}

else if (s.equals("close")) {

f.setVisible(false);

}

}

// Main class

public static void main(String args[])

{

List<String> words = List.of("hello", "dog", "hell", "cat", "a", "hel","help","helps","helping");

Autocompletion trie = new Autocompletion(words);

System.out.println(trie.suggest("st"));

new editor();

}

}

//A autocompletion class to autoconplete the user given word and poke th

//user the suggestions of the word;

public class Autocompletion {

public class TrieNode {

Map<Character, TrieNode> children;

char c;

boolean isWord;

public TrieNode(char c) {

this.c = c;

children = new HashMap<>();

}

public TrieNode() {

children = new HashMap<>();

}

public void insert(String word) {

if (word == null || word.isEmpty())

return;

char firstChar = word.charAt(0);

TrieNode child = children.get(firstChar);

if (child == null) {

child = new TrieNode(firstChar);

children.put(firstChar, child);

}

if (word.length() > 1)

child.insert(word.substring(1));

else

child.isWord = true;

}

}

TrieNode root;

public Autocompletion(List<String> words) {

root = new TrieNode();

for (String word : words)

root.insert(word);

}

public boolean find(String prefix, boolean exact) {

TrieNode lastNode = root;

for (char c : prefix.toCharArray()) {

lastNode = lastNode.children.get(c);

if (lastNode == null)

return false;

}

return !exact || lastNode.isWord;

}

public boolean find(String prefix) {

return find(prefix, false);

}

public void suggestHelper(TrieNode root, List<String> list, StringBuffer curr) {

if (root.isWord) {

list.add(curr.toString());

}

if (root.children == null || root.children.isEmpty())

return;

for (TrieNode child : root.children.values()) {

suggestHelper(child, list, curr.append(child.c));

curr.setLength(curr.length() - 1);

}

}

public List<String> suggest(String prefix) {

List<String> list = new ArrayList<>();

TrieNode lastNode = root;

StringBuffer curr = new StringBuffer();

for (char c : prefix.toCharArray()) {

lastNode = lastNode.children.get(c);

if (lastNode == null)

return list;

curr.append(c);

}

suggestHelper(lastNode, list, curr);

return list;

}

}

**5.RESULT:**

Screen shot of output

**6. CONCLUSION:**

**7.APPLICATIONS:**

**8. FUTURE SCOPE:**

**9. COPY RIGHT AFFIRMATION:**

We undersigned pledge and represent that the source code printed in this mini project report does not violate any proprietary or personal rights of others (including, without limitation, any copyrights or privacy rights); that the Work is factually accurate and contains no matter libellous or otherwise unlawful; that we have substantially participated in the creation of the Work and that it represents our original work sufficient for us to claim authorship.

**Name of students Sign**

**1.**

**2.**

**3.**