PROGRAMMIMG IN JAVA

“TEXT EDITOR”

Bachelor in Technology

(Computer science and engineering)

SCHOOL OF CSE (LPU)

PHAGWARA

GROUP MEMBER

1. SUBHADIP DUTTA	**REG NO:** 12104729	**ROLLNO:**RK21BGB43
2. SUBHAM RAY	**REG NO:**12110037	**ROLLNO:**RK21BGB31
3. ANIWESH OJHA	**REG NO:**12111612	**ROLL NO:**RK21BGA37
4. NV ANURAG	**REG NO:**12104883	**ROLL NO:**RK21BGA25

SECTION: K21BG

Course code: CSE310

Faculty: SHRUTI MAM

LOVELY PROFESSIONAL UNIVERSITY

Phagwara (Punjab)

TABLE OF CONTENT

INTRODUCTION

A text editor is a tool that allows a user to create and revise documents in a computer. Though this task can be carried out in other modes, the word text editor commonly refers to the tool that does this interactively.
The structure of a text editor depends largely on the types of editing features and displaying capabilities that are to be supported. To implement the diplaying capabilities, the semantics of the meta data that may be present in the document file needs to be implemented as display actions.

A text editor is program that **allows you to open, view, and edit plain text files**. Unlike word processors, text editors do not add formatting to text, instead focusing on editing functions for plain text. Text editors are used by a wide variety of people, for a wide variety of purposes.

PROJECT OVERVIEW

To create a simple text editor in Java Swing we will use a JTextArea, a JMenuBar and add JMenu to it and we will add JMenuItems. All the menu items will have actionListener to detect any action.
There will be a menu bar and it will contain two menus and a button:

1. **File menu**
    - **open**: this menuitem is used to open a file
    - **save**: this menuitem is used to save a file
    - **print** : this menuitem is used to print the components of the text area
    - **new** : this menuitem is used to create a new blank file
2. **Edit menu**
    - **cut**: this menuitem is to cut the selected area and copy it to clipboard
    - **copy**: this menuitem is to copy the selected area to the clipboard
    - **paste** : this menuitem is to paste the text from the clipboard to the text area
3. **Close** : this button closes the frame

method	explanation
cut()	removes the selected area from the text area and store it in clipboard
copy()	copies the selected area from the text area and store it in clipboard
paste()	removes the selected area from the text area and store it in clipboard
print()	prints the components of the text area

NEED OF TEXT EDITOR

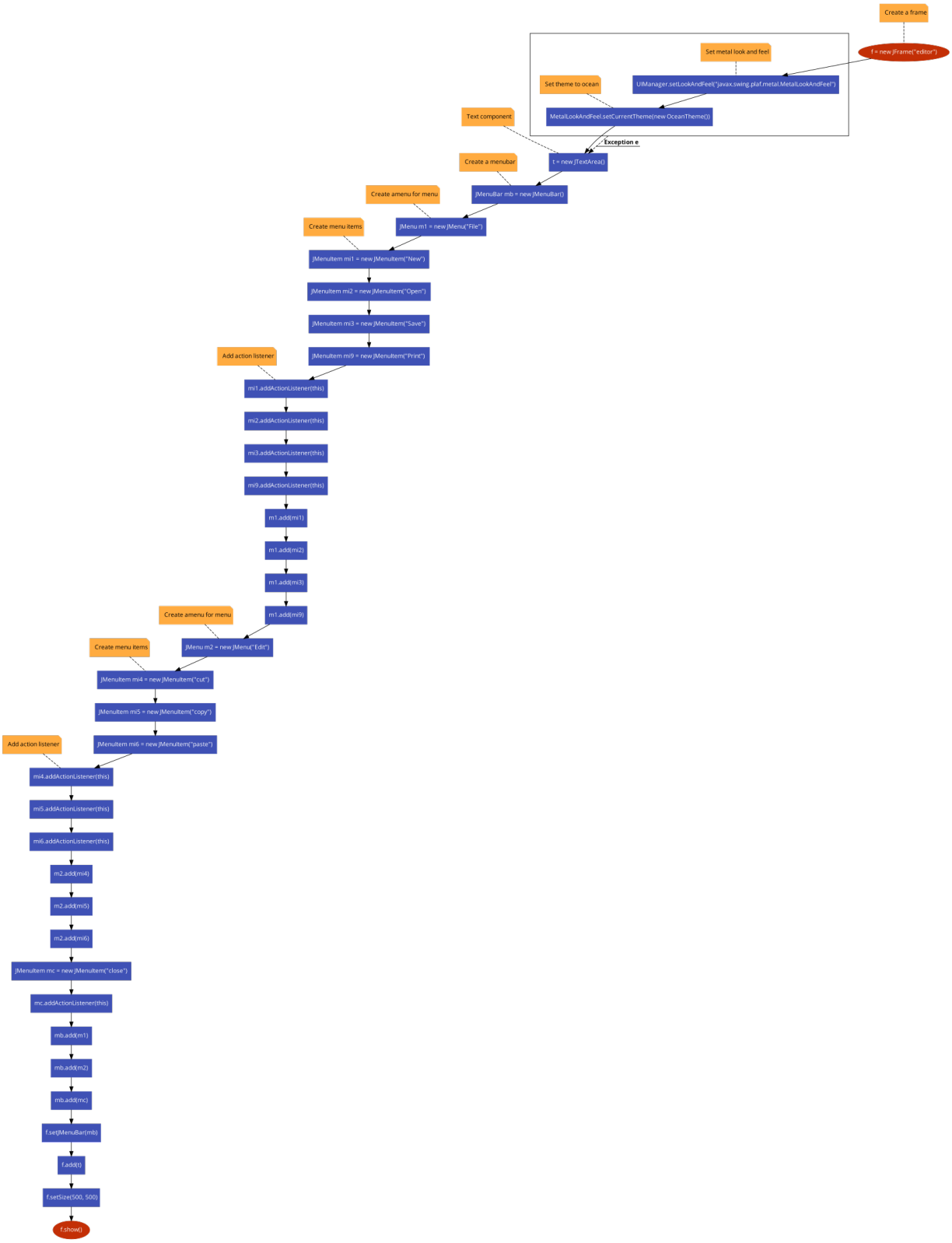
A text editor is a computer program that enables users to create, change, or edit plain text files (i.e., files with the suffix *.txt*). They're often used to craft complex code for websites; read, create and edit source code; or build text files.

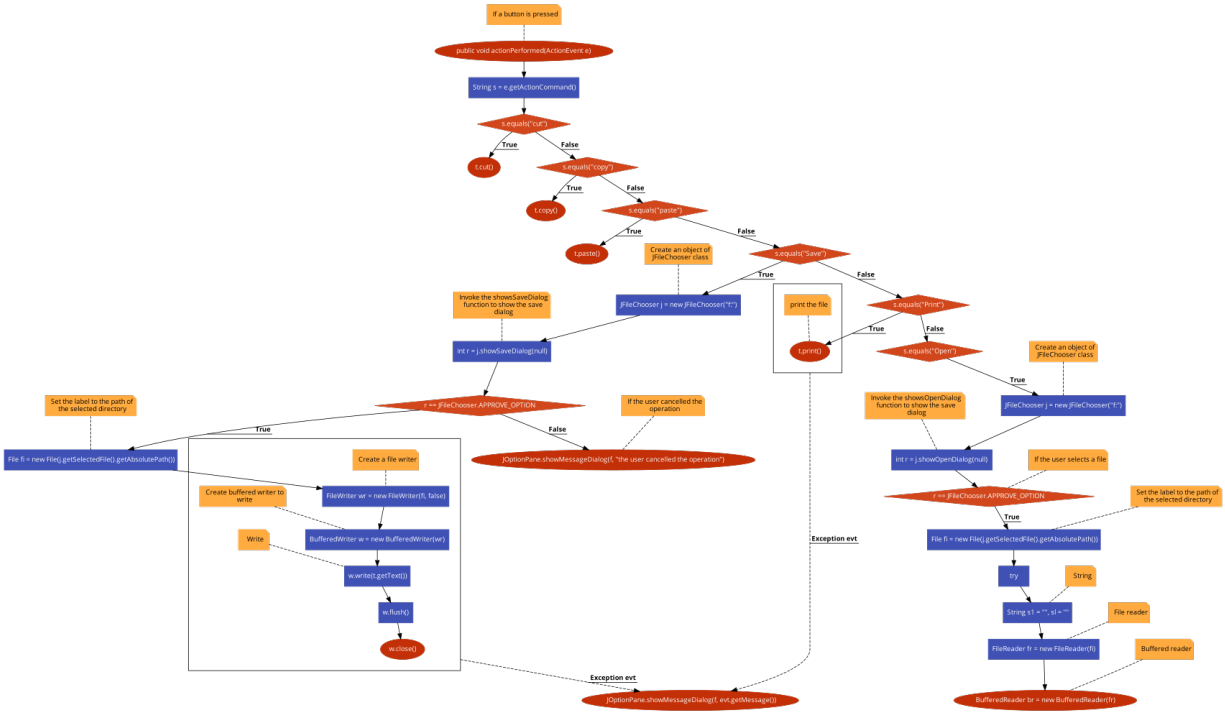
1. It improves user experience by going beyond regular text editing.

2. It has a place on most websites and web apps

3. It allows non-technical users to build pages or make content with little or no coding

FLOWCHART AND ALGORITHM





1. Import the necessary packages:
    - java.awt.*
    - javax.swing.*
    - java.io.*
    - java.awt.event.*
    - javax.swing.plaf.metal.*
2. Create a class named `editor` that extends `JFrame` and implements `ActionListener`.
3. Declare the following variables:
    - `JTextArea t` for the text component.
    - `JFrame f` for the frame.
4. Create a constructor named `editor`:
    - Instantiate the frame object.
    - Set the look and feel of the frame using `setLookAndFeel()`.
    - Create a `JTextArea` object.
    - Create a `JMenuBar` object.
    - Create a `JMenu` object for the menu.
    - Create menu items for the menu.

- Add action listener to each menu item.
- Add menu items to the menu.
- Add a `JMenuItem` object for the "close" option.
- Add action listener to the "close" option.
- Add the menu to the menu bar.
- Set the menu bar to the frame.
- Add the text component to the frame.
- Set the size of the frame.
- Make the frame visible.

5. Implement the `actionPerformed` method:
    - Get the action command of the event.
    - If the action command is "cut", "copy", or "paste", call the corresponding method of the text component.
    - If the action command is "Save", open a file chooser dialog to select the location to save the file. Create a `FileWriter` object and a `BufferedWriter` object to write the content of the text component to the selected file.
    - If the action command is "Print", print the content of the text component.
    - If the action command is "Open", open a file chooser dialog to select the file to open. Create a `FileReader` object and a `BufferedReader` object to read the content of the selected file and set the text component with the content.
    - If the action command is "New", clear the text component.
    - If the action command is "close", close the frame.
6. Create a `main` method:
    - Instantiate the `editor` object.
7. End of the algorithm.

CODE SNIPPET

Create a frame

```
        f = new JFrame("editor");

        try {
            // Set metal look and feel
            UIManager.setLookAndFeel("javax.swing.plaf.meta
l.MetalLookAn");

            // Set theme to ocean
            MetalLookAndFeel.setCurrentTheme(new
OceanTheme());
        }
        catch (Exception e) {
        }
```

Create a menubar

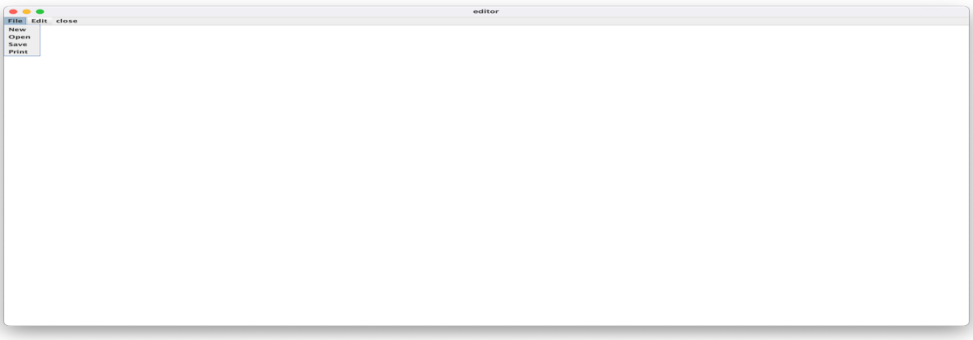
```
      JMenuBar mb = new JMenuBar();
```

Create amenu 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");
```

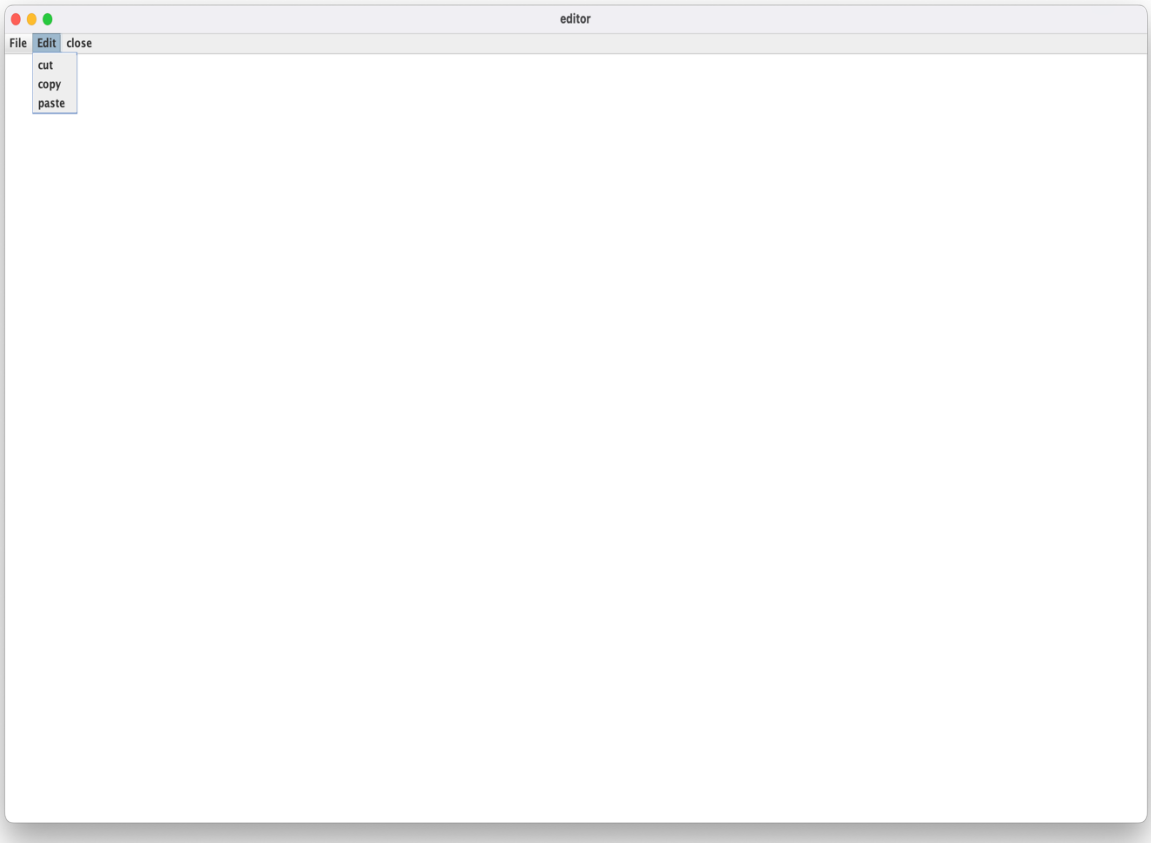


Add action listener

```
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 menu items

```
JMenuItem mi4 = new JMenuItem("cut");
JMenuItem mi5 = new JMenuItem("copy");
JMenuItem mi6 = new JMenuItem("paste");
```

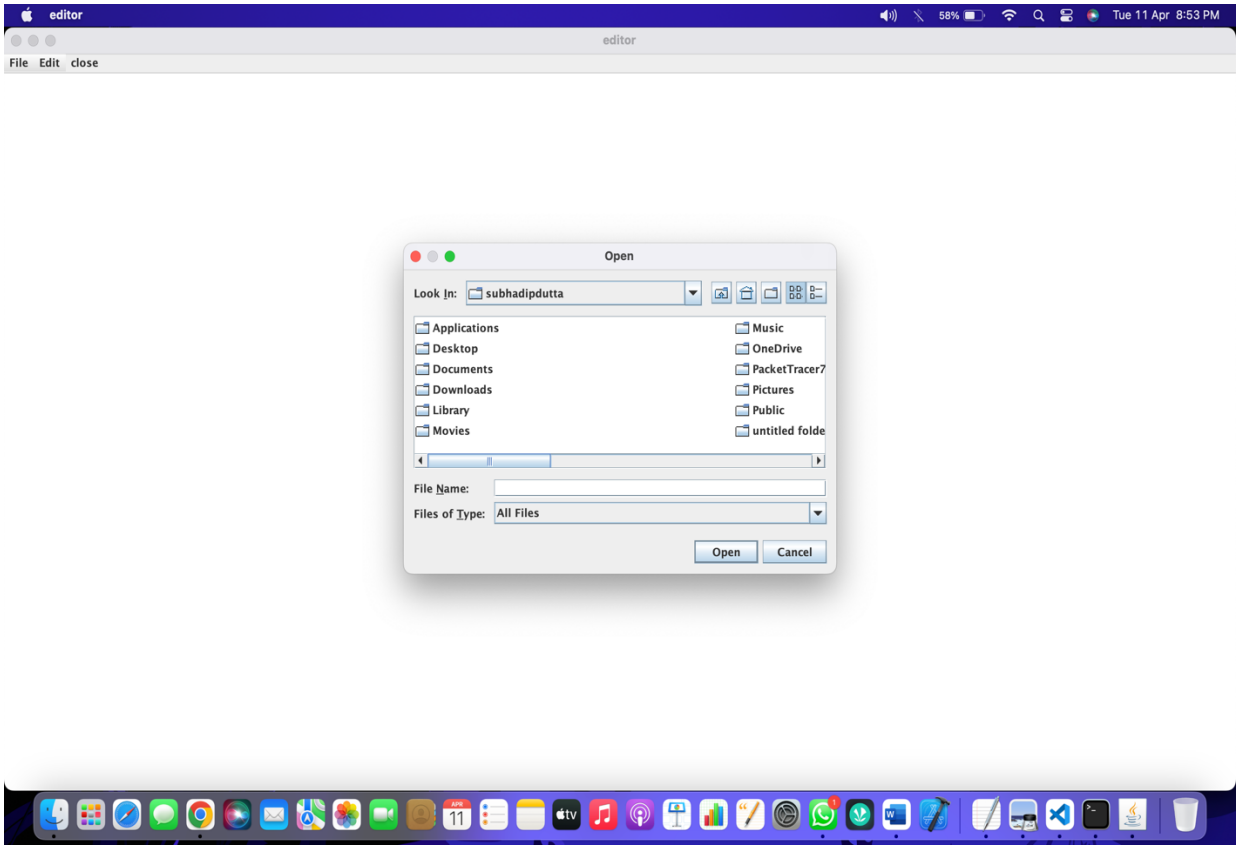


Add action listener

```
        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();
      }
```

If a button is pressed

```
      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")) {
```



FULL CODE WITH ALL MODULE:-

```
// Java Program to create a text editor using java
import java.awt.*;
import javax.swing.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.plaf.metal.*;
import javax.swing.text.*;
class editor extends JFrame implements ActionListener {
        // Text component
        JTextArea t;

        // Frame
        JFrame f;

        // Constructor
        editor()
```

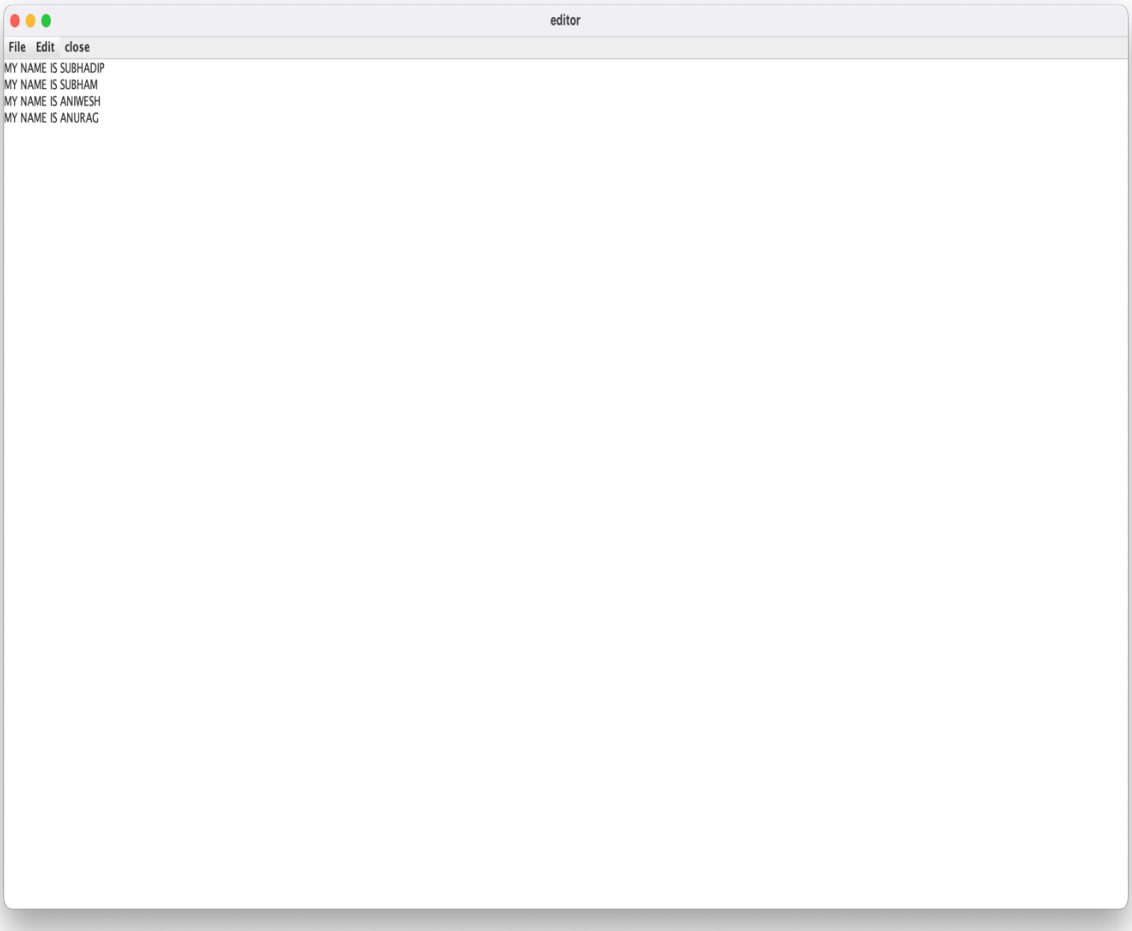
```
{
        // Create a frame
        f = new JFrame("editor");

        try {
                // Set metal look and feel

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

                // Set theme to ocean
                MetalLookAndFeel.setCurrentTheme(new OceanTheme());
        }
        catch (Exception e) {
        }

        // Text component
        t = new JTextArea();

        // Create a menubar
        JMenuBar mb = new JMenuBar();

        // Create amenu 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);
```

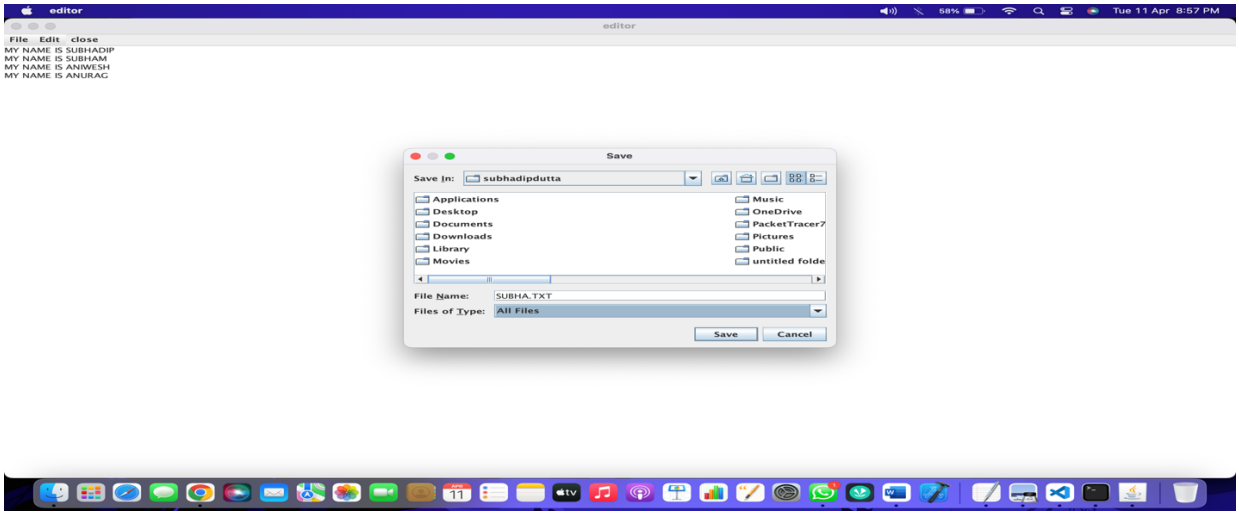
```
        m1.add(mi9);

        // 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");

        // Add action listener
        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();
}

// If a button is pressed
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")) {
                // Create an object of JFileChooser class
                JFileChooser j = new JFileChooser("f:");

                // Invoke the showsSaveDialog function to show the save
dialog
                int r = j.showSaveDialog(null);

                if (r == JFileChooser.APPROVE_OPTION) {

                    // Set the label to the path of the selected directory
                    File fi = new
File(j.getSelectedFile().getAbsolutePath());

                    try {
                        // Create a file writer
                        FileWriter wr = new FileWriter(fi, false);

                        // Create buffered writer to write
                        BufferedWriter w = new BufferedWriter(wr);

                        // Write
                        w.write(t.getText());

                        w.flush();
                        w.close();
                    }
                    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("Print")) {
                try {
                        // print the file
                        t.print();
                }
                catch (Exception evt) {
                        JOptionPane.showMessageDialog(f, evt.getMessage());
                }
        }
        else if (s.equals("Open")) {
                // Create an object of JFileChooser class
                JFileChooser j = new JFileChooser("f:");

                // Invoke the showsOpenDialog function to show the save
dialog
                int r = j.showOpenDialog(null);

                // If the user selects a file
                if (r == JFileChooser.APPROVE_OPTION) {
                        // Set the label to the path of the selected directory
                        File fi = new
File(j.getSelectedFile().getAbsolutePath());

                        try {
                                // String
                                String s1 = "", sl = "";

                                // File reader
                                FileReader fr = new FileReader(fi);

                                // 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[])
        {
                editor e = new editor();
        }
}
```


editor
File Edit close
MY NAME IS SUBHADIP
MY NAME IS SUBHAM
MY NAME IS ANIWESH
MY NAME IS ANURAG


editor
Tue 11 Apr 8:57 PM
editor
File Edit close
MY NAME IS SUBHADIP
MY NAME IS SUBHAM
MY NAME IS ANIWESH
MY NAME IS ANURAG
Save
Save In: subhadipdutta
Applications
Desktop
Documents
Downloads
Library
Movies
Music
OneDrive
PacketTracer7
Pictures
Public
untitled folde
File Name: SUBHA.TXT
Files of Type: All Files
Save
Cancel

RESULT AND DISCUSSION

The Text Editor Java Project was a success. We were able to successfully implement a Text Editor using Java, which allows a user to create and revise documents in a computer. Though this task can be carried out in other modes, the word text editor commonly refers to the tool that does this interactively.

Copy, paste, and cut, along with finding and replacing words and creating bulleted lists, are typical across text editor platforms.

Whereas the output of pure text editors is **plain text only**, word processors can also produce output in at least one binary format, and often in a number of binary formats. A binary format is one in which at least part of the data is in non-plain text form.

FUTURE SCOPE

The text editor Java project has a vast potential for future improvements and enhancements. Here are some potential future scopes for the project:

1. A text editor is program that **allows you to open, view, and edit plain text files**. Unlike word processors, text editors do not add formatting to text, instead focusing on editing functions for plain text. Text editors are used by a wide variety of people, for a wide variety of purposes.

2. Content editors are responsible for **researching, proofreading, and publishing both traditional and online media**. They analyze readership data and develop content strategies to increase user engagement. Content editors also fact-check articles and ensure the use of proper spelling, grammar, and syntax in outputs.

3. Text formatting – Text editors often provide basic visual formatting features like **line wrap, auto-indentation, bullet list formatting using ASCII characters, comment formatting, syntax highlighting** and so on.

CONCLUSION

In conclusion, text editor Java project can be a challenging but rewarding endeavor. By following the proposed methodology, the project can achieve high levels of accuracy in identifying faces, making it useful for various applications, such as security, access control, and human-computer interaction.

The project's future scope is significant, with opportunities to improve accuracy, speed, and security using advanced technologies such as deep learning, multi-modal recognition, and cloud-based face recognition. These enhancements can provide more effective and efficient solutions for real-world problems.

However, the project also has its challenges and limitations, such as dealing with low-quality features.

Overall, a text editor project can provide a foundation for building more advanced and sophisticated systems in the future.

ENDING AND ACKNOWLEDGEMENTS

In conclusion, we would like to express our sincere gratitude to you, **Shruti Ma'am,** for guiding us throughout the development of this text editor Java project. Your unwavering support and encouragement helped us navigate the challenges and complexities of the project, and your valuable feedback and suggestions greatly contributed to its success.

We would also like to extend our appreciation to the opensource community and the developers whose work made it possible for us to utilize a wide range of libraries and frameworks to develop the project. We would like to thank our colleagues and friends who provided their support and feedback during the development process.

Once again, we thank you for your invaluable support and guidance throughout this project. It was a privilege to work under your supervision.