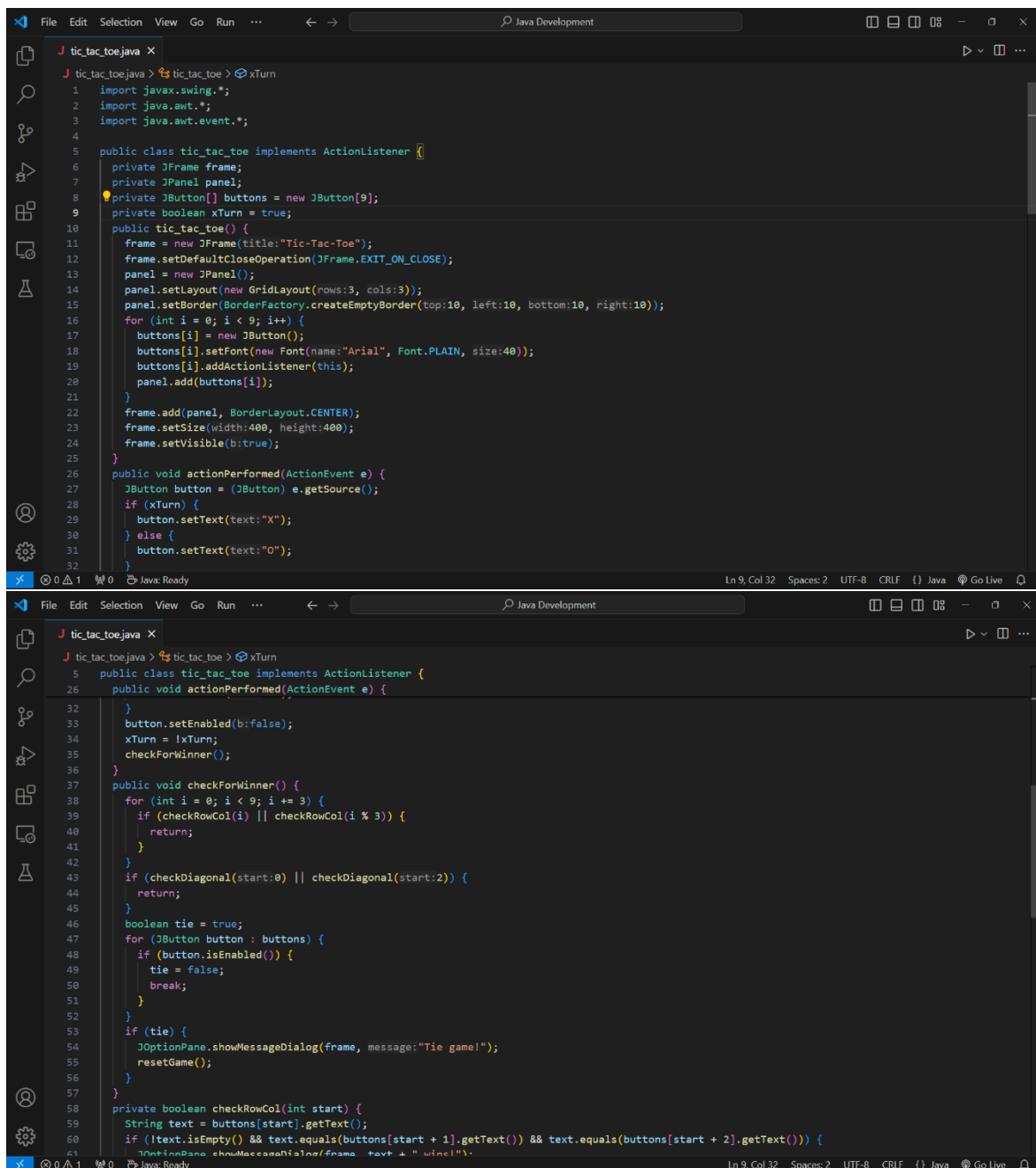


ASSINGMENT-4

APPLICATIONS

- SUBMITTED BY: - KANISHK
- CLASS: - CSE-37
- ROLL NO: - 2205130

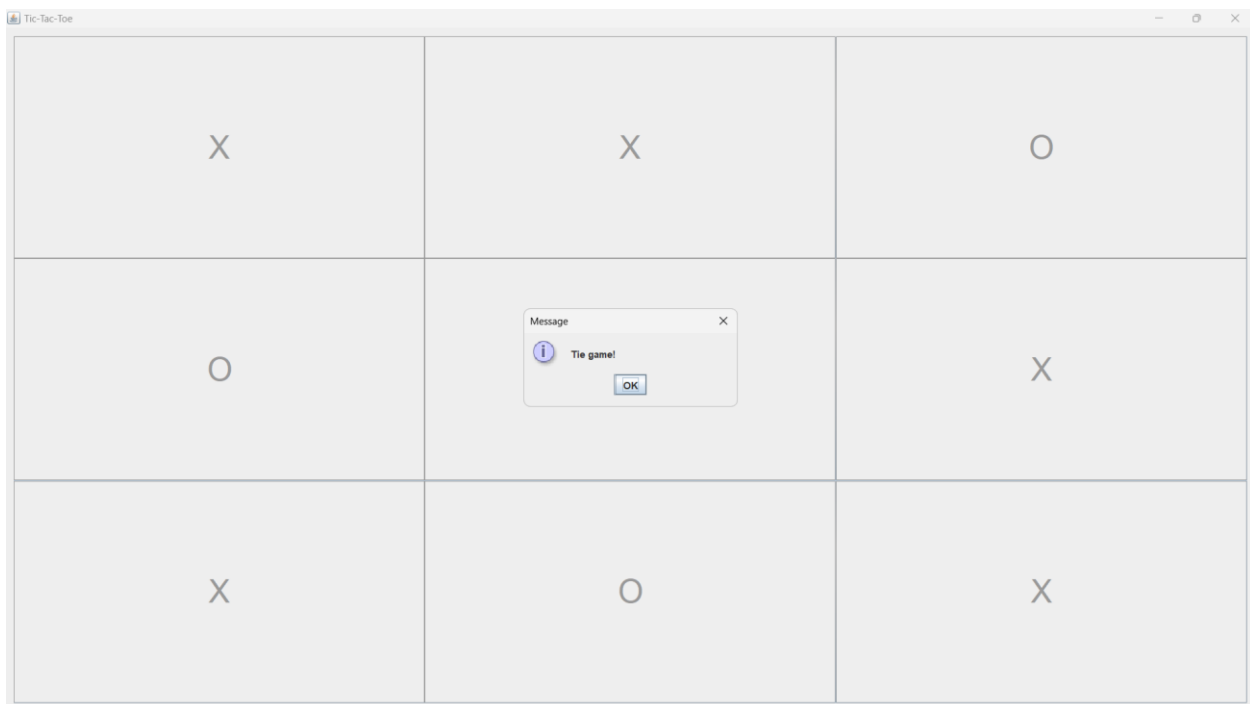
1. NOTEPAD INPUT: -



```
1  tic_tac_toe.java X
2  J tic_tac_toe.java > tic_tac_toe > xTurn
3  import javax.swing.*;
4  import java.awt.*;
5  import java.awt.event.*;
6
7  public class tic_tac_toe implements ActionListener {
8  private JFrame frame;
9  private JPanel panel;
10 private JButton[] buttons = new JButton[9];
11 private boolean xTurn = true;
12
13 public tic_tac_toe() {
14     frame = new JFrame(title:"Tic-Tac-Toe");
15     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16     panel = new JPanel();
17     panel.setLayout(new GridLayout(rows:3, cols:3));
18     panel.setBorder(BorderFactory.createEmptyBorder(top:10, left:10, bottom:10, right:10));
19     for (int i = 0; i < 9; i++) {
20         buttons[i] = new JButton();
21         buttons[i].setFont(new Font(name:"Arial", Font.PLAIN, size:40));
22         buttons[i].addActionListener(this);
23         panel.add(buttons[i]);
24     }
25     frame.add(panel, BorderLayout.CENTER);
26     frame.setSize(width:400, height:400);
27     frame.setVisible(b:true);
28 }
29
30 public void actionPerformed(ActionEvent e) {
31     JButton button = (JButton) e.getSource();
32     if (xTurn) {
33         button.setText(text:"X");
34     } else {
35         button.setText(text:"O");
36     }
37 }
38
39 public void checkForWinner() {
40     for (int i = 0; i < 9; i += 3) {
41         if (checkRowCol(i) || checkRowCol(i % 3)) {
42             return;
43         }
44     }
45     if (checkDiagonal(start:0) || checkDiagonal(start:2)) {
46         return;
47     }
48     boolean tie = true;
49     for (JButton button : buttons) {
50         if (button.isEnabled()) {
51             tie = false;
52             break;
53         }
54     }
55     if (tie) {
56         JOptionPane.showMessageDialog(frame, message:"Tie game!");
57         resetGame();
58     }
59 }
60
61 private boolean checkRowCol(int start) {
62     String text = buttons[start].getText();
63     if (!text.isEmpty() && text.equals(buttons[start + 1].getText()) && text.equals(buttons[start + 2].getText())) {
64         JOptionPane.showMessageDialog(frame, text + " wins!");
65     }
66 }
```

```
File Edit Selection View Go Run ... Java Development
J tic_tac_toe.java X
J tic_tac_toe.java > ? tic_tac_toe > xTurn
5 public class tic_tac_toe implements ActionListener {
58 private boolean checkRowCol(int start) {
63     return true;
64 }
65     return false;
66 }
67 private boolean checkDiagonal(int start) {
68     String text = buttons[start].getText();
69     if (!text.isEmpty() && text.equals(buttons[4].getText()) && text.equals(buttons[start == 0 ? 8 : 6].getText())) {
70         JOptionPane.showMessageDialog(frame, text + " wins!");
71         resetGame();
72         return true;
73     }
74     return false;
75 }
76 public void resetGame() {
77     for (JButton button : buttons) {
78         button.setText(text:"");
79         button.setEnabled(b:true);
80     }
81     xTurn = true;
82 }
83 Run | Debug
84 public static void main(String[] args) {
85     new tic_tac_toe();
86 }
Ln 9, Col 32 Spaces: 2 UTF-8 CRLF {} Java Go Live
```

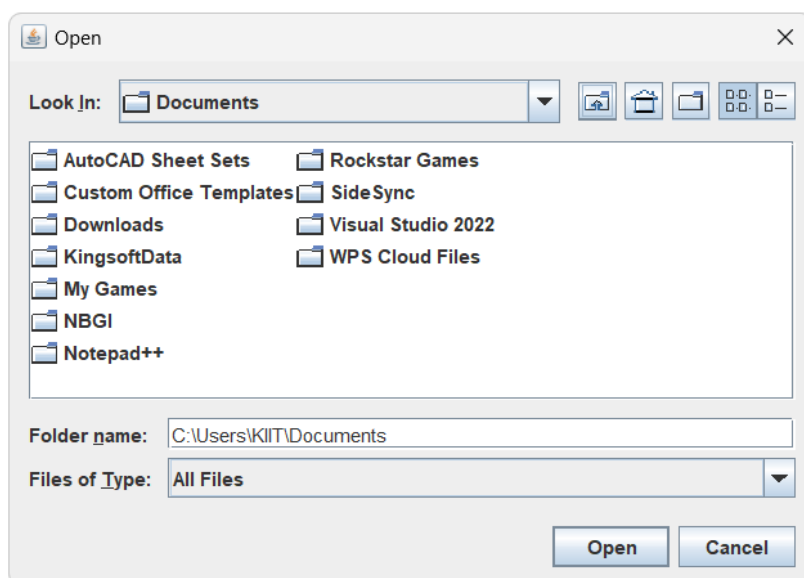
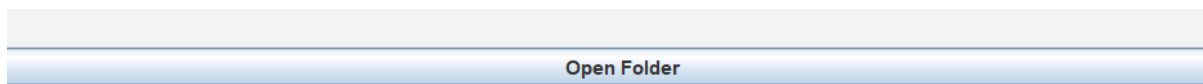
OUTPUT: -



2.FOLDER_OPENER INPUT: -

```
File Edit Selection View Go Run Terminal Help Java Development
J tic_tac_toe.java J FolderOpener[1].java 1
C:\Users\KIIT> AppData\Local\Microsoft\Windows\INetCache\IE\UGIE2F5Z> J FolderOpener[1].java > ...
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.*;
4 import java.io.File;
5
6 public class FolderOpener extends JFrame implements ActionListener {
7     JButton openButton;
8     JList<String> listView;
9     public FolderOpener() {
10         super(title:"Folder Opener");
11         openButton = new JButton(text:"Open Folder");
12         listView = new JList<>();
13         openButton.addActionListener(this);
14         JPanel panel = new JPanel();
15         panel.setLayout(new BorderLayout());
16         panel.add(openButton, BorderLayout.NORTH);
17         panel.add(new JScrollPane(listView), BorderLayout.CENTER);
18         add(panel);
19         pack();
20         setVisible(true);
21     }
22     public void actionPerformed(ActionEvent e) {
23         if (e.getSource() == openButton) {
24             JFileChooser fileChooser = new JFileChooser();
25             fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
26             int returnVal = fileChooser.showOpenDialog(this);
27             if (returnVal == JFileChooser.APPROVE_OPTION) {
28                 File selectedFolder = fileChooser.getSelectedFile();
29                 File[] contents = selectedFolder.listFiles();
30                 String[] contentNames = new String[contents.length];
31                 for (int i = 0; i < contents.length; i++) {
32                     contentNames[i] = contents[i].getName();
33                 }
34                 listView.setListData(contentNames);
35             }
36         }
37     }
38     public static void main(String[] args) {
39         new FolderOpener();
40     }
41 }
```

OUTPUT: -



3.CALCULATOR_APP INPUT: -

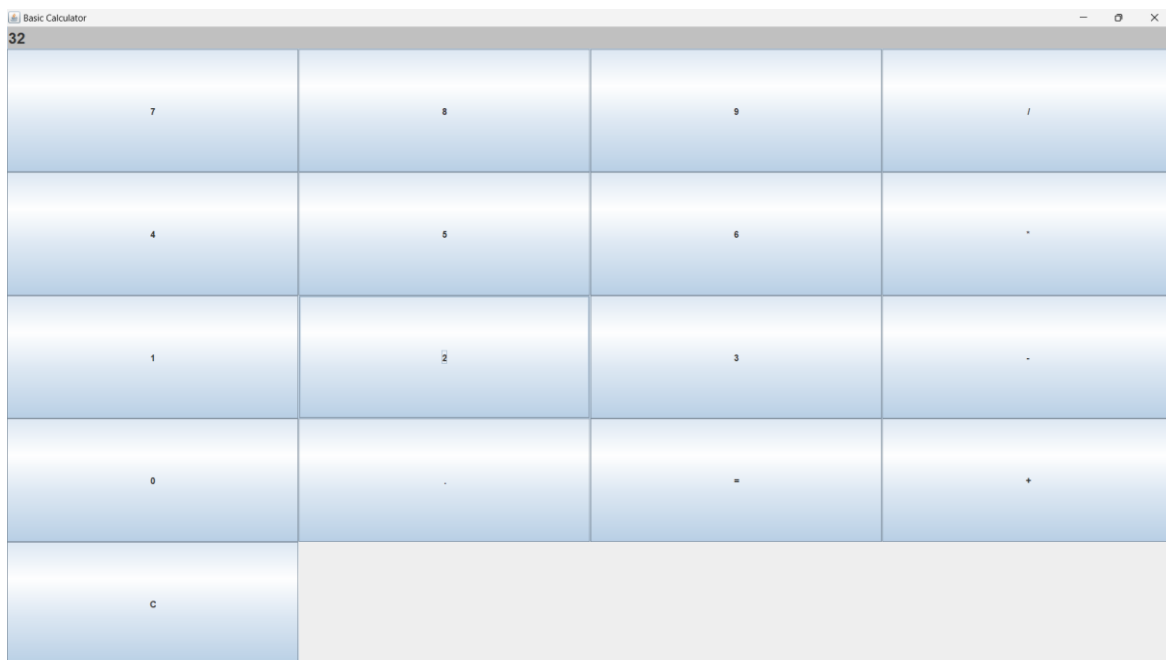
```
File Edit Selection View Go Run Terminal Help
CalculatorApp.java X
CalculatorApp.java > CalculatorApp > firstValue
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 public class CalculatorApp extends JFrame implements ActionListener {
7     private JTextField display;
8     private double firstValue = 0;
9     private String operator = "";
10    public CalculatorApp() {
11        super("Basic Calculator");
12        display = new JTextField();
13        display.setFont(new Font("Sans Serif", Font.BOLD, size:20));
14        display.setEditable(false);
15        display.setBackground(Color.LIGHT_GRAY);
16        JButton button7 = new JButton(text:"7");
17        JButton button8 = new JButton(text:"8");
18        JButton button9 = new JButton(text:"9");
19        JButton buttonDivide = new JButton(text:"/");
20        JButton button4 = new JButton(text:"4");
21        JButton button5 = new JButton(text:"5");
22        JButton button6 = new JButton(text:"6");
23        JButton buttonMultiply = new JButton(text:"*");
24        JButton button1 = new JButton(text:"1");
25        JButton button2 = new JButton(text:"2");
26        JButton button3 = new JButton(text:"3");
27        JButton buttonSubtract = new JButton(text:"-");
28        JButton button0 = new JButton(text:"0");
29        JButton buttonDecimal = new JButton(text:".");
30        JButton buttonEquals = new JButton(text:"=");
31        JButton buttonAdd = new JButton(text:"+");
32        JButton buttonClear = new JButton(text:"C");
33        button7.addActionListener(this);
34        button8.addActionListener(this);
35        button9.addActionListener(this);
36        buttonDivide.addActionListener(this);
37        button4.addActionListener(this);
38        button5.addActionListener(this);
39        button6.addActionListener(this);
40        buttonMultiply.addActionListener(this);
41        button1.addActionListener(this);
42        button2.addActionListener(this);
43        button3.addActionListener(this);
44        buttonSubtract.addActionListener(this);
45        button0.addActionListener(this);
46        buttonDecimal.addActionListener(this);
47        buttonEquals.addActionListener(this);
48        buttonAdd.addActionListener(this);
49        buttonClear.addActionListener(this);
50        JPanel buttonPanel = new JPanel(new GridLayout(rows:5, cols:4));
51        buttonPanel.add(button7);
52        buttonPanel.add(button8);
53        buttonPanel.add(button9);
54        buttonPanel.add(buttonDivide);
55        buttonPanel.add(button4);
56        buttonPanel.add(button5);
57        buttonPanel.add(button6);
58        buttonPanel.add(buttonMultiply);
59        buttonPanel.add(button1);
60        buttonPanel.add(button2);
61        buttonPanel.add(button3);
62        buttonPanel.add(buttonSubtract);
63        buttonPanel.add(button0);
64        buttonPanel.add(buttonDecimal);
65        buttonPanel.add(buttonEquals);
66        buttonPanel.add(buttonAdd);
67        buttonPanel.add(buttonClear);
68        getContentPane().add(display, BorderLayout.NORTH);
69        getContentPane().add(buttonPanel, BorderLayout.CENTER);
70        setSize(width:300, height:400);
71        setVisible(true);
72        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
73    }
74
75    @Override
76    public void actionPerformed(ActionEvent e) {
77        String command = e.getActionCommand();
78        if (Character.isDigit(command.charAt(index:0))) {
79            display.setText(display.getText() + command);
80        } else if (command.equals(operator:"=")) {
81            if (display.getText().contains(s:"/")) {
82                display.setText(display.getText() + ".");
83            }
84        } else if (command.equals(operator:"C")) {
85            display.setText("");
86            firstValue = 0;
87            operator = "";
88        } else {
89            operator = command;
90            firstValue = Double.parseDouble(display.getText());
91            display.setText("");
92        }
93        if (command.equals(operator:"=")) {
94            double secondValue = Double.parseDouble(display.getText());
95            double result = calculate(firstValue, operator, secondValue);
96            display.setText(String.valueOf(result));
97            firstValue = 0;
98            operator = "";
99        }
100    }
101
102    private double calculate(double firstValue, String operator, double secondValue) {
103        switch (operator) {
104            case "+":
105                return firstValue + secondValue;
106            case "-":
107                return firstValue - secondValue;
108            case "*":
109                return firstValue * secondValue;
110            case "/":
111                if (secondValue == 0) {
112                    JOptionPane.showMessageDialog(this, message:"Error: Cannot divide by zero!", title:"Calculator Error", JOptionPane.ERROR_MESSAGE);
113                }
114                return firstValue / secondValue;
115        }
116    }
117}
```

```
File Edit Selection View Go Run Terminal Help
CalculatorApp.java X
CalculatorApp.java > CalculatorApp > firstValue
40 buttonMultiply.addActionListener(this);
41 button1.addActionListener(this);
42 button2.addActionListener(this);
43 button3.addActionListener(this);
44 buttonSubtract.addActionListener(this);
45 button0.addActionListener(this);
46 buttonDecimal.addActionListener(this);
47 buttonEquals.addActionListener(this);
48 buttonAdd.addActionListener(this);
49 buttonClear.addActionListener(this);
50 JPanel buttonPanel = new JPanel(new GridLayout(rows:5, cols:4));
51 buttonPanel.add(button7);
52 buttonPanel.add(button8);
53 buttonPanel.add(button9);
54 buttonPanel.add(buttonDivide);
55 buttonPanel.add(button4);
56 buttonPanel.add(button5);
57 buttonPanel.add(button6);
58 buttonPanel.add(buttonMultiply);
59 buttonPanel.add(button1);
60 buttonPanel.add(button2);
61 buttonPanel.add(button3);
62 buttonPanel.add(buttonSubtract);
63 buttonPanel.add(button0);
64 buttonPanel.add(buttonDecimal);
65 buttonPanel.add(buttonEquals);
66 buttonPanel.add(buttonAdd);
67 buttonPanel.add(buttonClear);
68 getContentPane().add(display, BorderLayout.NORTH);
69 getContentPane().add(buttonPanel, BorderLayout.CENTER);
70 setSize(width:300, height:400);
71 setVisible(true);
72 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
73
74
75 @Override
76 public void actionPerformed(ActionEvent e) {
77     String command = e.getActionCommand();
78     if (Character.isDigit(command.charAt(index:0))) {
79         display.setText(display.getText() + command);
80     } else if (command.equals(operator:"=")) {
81         if (display.getText().contains(s:"/")) {
82             display.setText(display.getText() + ".");
83         }
84     } else if (command.equals(operator:"C")) {
85         display.setText("");
86         firstValue = 0;
87         operator = "";
88     } else {
89         operator = command;
90         firstValue = Double.parseDouble(display.getText());
91         display.setText("");
92     }
93     if (command.equals(operator:"=")) {
94         double secondValue = Double.parseDouble(display.getText());
95         double result = calculate(firstValue, operator, secondValue);
96         display.setText(String.valueOf(result));
97         firstValue = 0;
98         operator = "";
99     }
100 }
101
102 private double calculate(double firstValue, String operator, double secondValue) {
103     switch (operator) {
104         case "+":
105             return firstValue + secondValue;
106         case "-":
107             return firstValue - secondValue;
108         case "*":
109             return firstValue * secondValue;
110         case "/":
111             if (secondValue == 0) {
112                 JOptionPane.showMessageDialog(this, message:"Error: Cannot divide by zero!", title:"Calculator Error", JOptionPane.ERROR_MESSAGE);
113             }
114             return firstValue / secondValue;
115     }
116 }
```

```
File Edit Selection View Go Run Terminal Help
CalculatorApp.java X
CalculatorApp.java > CalculatorApp > firstValue
74
75 @Override
76 public void actionPerformed(ActionEvent e) {
77     String command = e.getActionCommand();
78     if (Character.isDigit(command.charAt(index:0))) {
79         display.setText(display.getText() + command);
80     } else if (command.equals(operator:"=")) {
81         if (display.getText().contains(s:"/")) {
82             display.setText(display.getText() + ".");
83         }
84     } else if (command.equals(operator:"C")) {
85         display.setText("");
86         firstValue = 0;
87         operator = "";
88     } else {
89         operator = command;
90         firstValue = Double.parseDouble(display.getText());
91         display.setText("");
92     }
93     if (command.equals(operator:"=")) {
94         double secondValue = Double.parseDouble(display.getText());
95         double result = calculate(firstValue, operator, secondValue);
96         display.setText(String.valueOf(result));
97         firstValue = 0;
98         operator = "";
99     }
100 }
101
102 private double calculate(double firstValue, String operator, double secondValue) {
103     switch (operator) {
104         case "+":
105             return firstValue + secondValue;
106         case "-":
107             return firstValue - secondValue;
108         case "*":
109             return firstValue * secondValue;
110         case "/":
111             if (secondValue == 0) {
112                 JOptionPane.showMessageDialog(this, message:"Error: Cannot divide by zero!", title:"Calculator Error", JOptionPane.ERROR_MESSAGE);
113             }
114             return firstValue / secondValue;
115     }
116 }
```

```
CalculatorApp.java X
J CalculatorApp.java > CalculatorApp > firstValue
6 public class CalculatorApp extends JFrame implements ActionListener {
100 private double calculate(double firstValue, String operator, double secondValue) {
106     case "+":
107         return firstValue * secondValue;
108     case "/":
109         if (secondValue == 0) {
110             JOptionPane.showMessageDialog(this, message:"Error: Cannot divide by zero!", title:"Calculator Error", JOptionPane.ERROR_MESSAGE);
111             return 0;
112         } else {
113             return firstValue / secondValue;
114         }
115     default:
116         return 0;
117     }
118 }
119 Run | Debug
120 public static void main(String[] args) {
121     new CalculatorApp();
122 }
123 }
```

OUTPUT: -

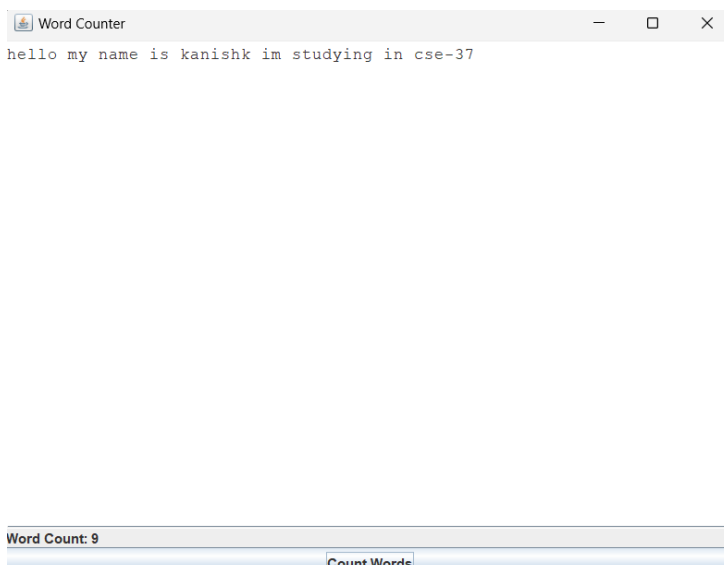


4. WORD_COUNTER_APP. INPUT: -

```
File Edit Selection View Go Run ... Java Development
J WordCounterApp.java x
J WordCounterApp.java > WordCounterApp > WordCounterApp()
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 public class WordCounterApp extends JFrame implements ActionListener {
7     private JTextArea inputArea;
8     private JLabel wordCountLabel;
9     private JButton countButton;
10    public WordCounterApp() {
11        super(title:"Word Counter");
12        inputArea = new JTextArea();
13        inputArea.setFont(new Font(name:"Monospaced", Font.PLAIN, size:14));
14        JScrollPane scrollPane = new JScrollPane(inputArea);
15        wordCountLabel = new JLabel(text:"Word Count: 0");
16        countButton = new JButton(text:"Count Words");
17        countButton.addActionListener(this);
18        JPanel contentPanel = new JPanel();
19        contentPanel.setLayout(new BorderLayout());
20        contentPanel.add(scrollPane, BorderLayout.CENTER);
21        contentPanel.add(wordCountLabel, BorderLayout.SOUTH);
22        getContentPane().add(contentPanel, BorderLayout.CENTER);
23        getContentPane().add(countButton, BorderLayout.SOUTH);
24        setSize(width:500, height:300);
25        setVisible(b:true);
26        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27    }
28    @Override
29    public void actionPerformed(ActionEvent e) {
30        if (e.getSource() == countButton) {
31            String text = inputArea.getText();
32            String[] words = text.split(regex:"\\s+");
33        }
34    }
35
36    Run [Debug]
37    public static void main(String[] args) {
38        new WordCounterApp();
39    }
40
41
Ln 20, Col 52 Spaces: 4 UTF-8 CRLF {} Java Go Live
```

```
File Edit Selection View Go Run ... Java Development
J WordCounterApp.java x
J WordCounterApp.java > WordCounterApp > WordCounterApp()
6 public class WordCounterApp extends JFrame implements ActionListener {
10    public WordCounterApp() {
11    }
12
13    @Override
14    public void actionPerformed(ActionEvent e) {
15        if (e.getSource() == countButton) {
16            String text = inputArea.getText();
17            String[] words = text.split(regex:"\\s+");
18            int wordCount = words.length;
19            wordCountLabel.setText("Word Count: " + wordCount);
20        }
21    }
22
23    Run [Debug]
24    public static void main(String[] args) {
25        new WordCounterApp();
26    }
27
28
Ln 20, Col 52 Spaces: 4 UTF-8 CRLF {} Java Go Live
```

OUTPUT: -



5.SIMPLE NOTEPAD INPUT: -

```

1  SimpleNotepad.java | X
2  SimpleNotepad.java | SimpleNotepad > main(String[])
3      import java.awt.*;
4      import java.awt.event.*;
5      import java.io.BufferedReader;
6      import java.io.File;
7      import java.io.FileReader;
8      import java.io.FileWriter;
9      import javax.swing.*;
10     import javax.swing.event.*;
11     import javax.swing.filechooser.FileNameExtensionFilter;
12     import java.io.IOException;
13
14     public class SimpleNotepad extends JFrame implements ActionListener {
15
16         Run(Debug
17         public static void main(String[] args) {
18             new SimpleNotepad();
19         }
20
21         JTextArea textArea;
22         JMenuBar menuBar;
23         JMenu fileMenu, editMenu;
24         JMenuItem newMenuItem, openMenuItem, saveMenuItem, exitMenuItem, findMenuItem, replaceMenuItem, fontMenuItem;
25
26         String currentFile = "";
27         public SimpleNotepad() {
28             super("Simple Notepad");
29             textArea = new JTextArea();
30             textArea.setFont(new Font("Arial", Font.PLAIN, 16));
31             menuBar = new JMenuBar();
32             setMenuBar(menuBar);
33             fileMenu = new JMenu("File");
34             menuBar.add(fileMenu);
35             newMenuItem = new JMenuItem("New");
36             newMenuItem.addActionListener(this);
37             fileMenu.add(newMenuItem);
38             openMenuItem = new JMenuItem("Open");
39             openMenuItem.addActionListener(this);
40             fileMenu.add(openMenuItem);
41             saveMenuItem = new JMenuItem("Save");
42             saveMenuItem.addActionListener(this);
43             fileMenu.add(saveMenuItem);
44             fileMenu.addSeparator();
45             exitMenuItem = new JMenuItem("Exit");
46             exitMenuItem.addActionListener(this);
47             fileMenu.add(exitMenuItem);
48             editMenu = new JMenu("Edit");
49             menuBar.add(editMenu);
50             findMenuItem = new JMenuItem("Find");
51             findMenuItem.addActionListener(this);
52             editMenu.add(findMenuItem);
53             replaceMenuItem = new JMenuItem("Replace");
54             editMenu.add(replaceMenuItem);
55         }
56
57         public void actionPerformed(ActionEvent e) {
58             JMenu source = (JMenu) e.getSource();
59             if (source == newMenuItem) {
60                 textArea.setText("");
61                 currentFile = "";
62             } else if (source == openMenuItem) {
63                 openFile();
64             } else if (source == saveMenuItem) {
65                 saveFile();
66             } else if (source == exitMenuItem) {
67                 System.exit(0);
68             } else if (source == findMenuItem) {
69                 findText();
70             } else if (source == replaceMenuItem) {
71                 replaceText();
72             } else {
73             }
74         }
75
76         private void replaceText() {
77             String searchTerm = JOptionPane.showInputDialog(this, "Enter text to find:", "Replace", JOptionPane.PLAIN_MESSAGE);
78             String replaceTerm = JOptionPane.showInputDialog(this, "Enter text to replace with:", "Replace", JOptionPane.PLAIN_MESSAGE);
79             if (searchTerm != null && searchTerm.isEmpty() && replaceTerm != null) {
80                 int pos = 0;
81                 while ((pos = textArea.getText().indexOf(searchTerm, pos)) != -1) {
82                     textArea.replaceRange(replaceTerm, pos, pos + searchTerm.length());
83                     pos += replaceTerm.length();
84                 }
85                 JOptionPane.showMessageDialog(this, "Replace complete!", "Replace", JOptionPane.INFORMATION_MESSAGE);
86             }
87         }
88
89         private void findText() {
90             String searchTerm = JOptionPane.showInputDialog(this, "Enter text to find:", "Find", JOptionPane.PLAIN_MESSAGE);
91             if (searchTerm != null && searchTerm.isEmpty()) {
92                 int position = textArea.getText().indexOf(searchTerm);
93                 if (position != -1) {
94                     textArea.setSelectionStart(position);
95                     textArea.setSelectionEnd(position + searchTerm.length());
96                 } else {
97                     JOptionPane.showMessageDialog(this, "Text not found!", "Find", JOptionPane.INFORMATION_MESSAGE);
98                 }
99             }
100         }
101
102         private void openFile() {
103             JFileChooser fileChooser = new JFileChooser();
104             FileNameExtensionFilter filter = new FileNameExtensionFilter("Text Files", ".txt");
105             fileChooser.setFileFilter(filter);
106             int returnValue = fileChooser.showOpenDialog(this);
107             if (returnValue == JFileChooser.APPROVE_OPTION) {
108                 File selectedFile = fileChooser.getSelectedFile();
109                 currentFile = selectedFile.getAbsolutePath();
110                 try {
111                     BufferedReader reader = new BufferedReader(new FileReader(selectedFile));
112                     String line;
113                     while ((line = reader.readLine()) != null) {
114                         textArea.append(line + "\n");
115                     }
116                     reader.close();
117                 } catch (IOException ex) {
118                     JOptionPane.showMessageDialog(this, "Error opening file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
119                 }
120             }
121         }
122
123         private void saveFile() {
124             if (currentFile == null || currentFile.isEmpty()) {
125                 saveFileAs();
126             } else {
127                 try {
128                     FileWriter writer = new FileWriter(currentFile);
129                     textArea.write(writer, 0, textArea.getText().length());
130                     writer.close();
131                 } catch (IOException ex) {
132                     JOptionPane.showMessageDialog(this, "Error saving file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
133                 }
134             }
135         }
136
137         private void saveFileAs() {
138             JFileChooser fileChooser = new JFileChooser();
139             FileNameExtensionFilter filter = new FileNameExtensionFilter("Text Files", ".txt");
140             fileChooser.setFileFilter(filter);
141             int returnValue = fileChooser.showSaveDialog(this);
142             if (returnValue == JFileChooser.APPROVE_OPTION) {
143                 File selectedFile = fileChooser.getSelectedFile();
144                 currentFile = selectedFile.getAbsolutePath();
145                 try {
146                     FileWriter writer = new FileWriter(selectedFile);
147                     textArea.write(writer, 0, textArea.getText().length());
148                     writer.close();
149                 } catch (IOException ex) {
150                     JOptionPane.showMessageDialog(this, "Error saving file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
151                 }
152             }
153         }
154     }

```

```
12 public class SimpleNotepad extends JFrame implements ActionListener {
13     private void openFile() {
14         JFileChooser fileChooser = new JFileChooser();
15         fileChooser.setFileFilter(new FileFilter() {
16             public boolean accept(File f) {
17                 return f.isDirectory() || f.getName().toLowerCase().endsWith(".txt");
18             }
19         });
20         int returnVal = fileChooser.showOpenDialog(this);
21         if (returnVal == JFileChooser.APPROVE_OPTION) {
22             File file = fileChooser.getSelectedFile();
23             try {
24                 FileReader reader = new FileReader(file);
25                 BufferedReader br = new BufferedReader(reader);
26                 StringBuilder text = new StringBuilder();
27                 String line;
28                 while ((line = br.readLine()) != null) {
29                     text.append(line).append("\n");
30                 }
31                 br.close();
32                 reader.close();
33                 textArea.setText(text.toString());
34                 currentFile = file.getPath();
35             } catch (Exception ex) {
36                 ex.printStackTrace();
37                 JOptionPane.showMessageDialog(this, message: "Error opening file!", title: "Error", JOptionPane.ERROR_MESSAGE);
38             }
39         }
40     }
41     private void saveFile() {
42         if (currentFile.isEmpty()) {
43             saveAsFile();
44         } else {
45             try {
46                 FileWriter writer = new FileWriter(currentFile);
47                 BufferedWriter bw = new BufferedWriter(writer);
48                 bw.write(textArea.getText());
49                 bw.close();
50                 writer.close();
51                 JOptionPane.showMessageDialog(this, message: "File saved successfully!", title: "Success", JOptionPane.INFORMATION_MESSAGE);
52             } catch (Exception ex) {
53                 ex.printStackTrace();
54                 JOptionPane.showMessageDialog(this, message: "Error saving file!", title: "Error", JOptionPane.ERROR_MESSAGE);
55             }
56         }
57     }
58     private void saveAsFile() {
59         JFileChooser fileChooser = new JFileChooser();
60         fileChooser.setFileFilter(new FileFilter() {
61             public boolean accept(File f) {
62                 return f.isDirectory() || f.getName().toLowerCase().endsWith(".txt");
63             }
64         });
65         int returnVal = fileChooser.showSaveDialog(this);
66         if (returnVal == JFileChooser.APPROVE_OPTION) {
67             File file = fileChooser.getSelectedFile();
68             String filePath = file.getPath();
69             if (!filePath.toLowerCase().endsWith(".txt")) {
70                 filePath += ".txt";
71             }
72             try {
73                 FileWriter writer = new FileWriter(filePath);
74                 BufferedWriter bw = new BufferedWriter(writer);
75                 bw.write(textArea.getText());
76                 bw.close();
77                 writer.close();
78                 currentFile = filePath;
79                 JOptionPane.showMessageDialog(this, message: "File saved successfully!", title: "Success", JOptionPane.INFORMATION_MESSAGE);
80             } catch (Exception ex) {
81                 ex.printStackTrace();
82                 JOptionPane.showMessageDialog(this, message: "Error saving file!", title: "Error", JOptionPane.ERROR_MESSAGE);
83             }
84         }
85     }
86 }
```

```
12 public class SimpleNotepad extends JFrame implements ActionListener {
13     private void saveFile() {
14         JFileChooser fileChooser = new JFileChooser();
15         fileChooser.setFileFilter(new FileFilter() {
16             public boolean accept(File f) {
17                 return f.isDirectory() || f.getName().toLowerCase().endsWith(".txt");
18             }
19         });
20         int returnVal = fileChooser.showSaveDialog(this);
21         if (returnVal == JFileChooser.APPROVE_OPTION) {
22             File file = fileChooser.getSelectedFile();
23             String filePath = file.getPath();
24             if (!filePath.toLowerCase().endsWith(".txt")) {
25                 filePath += ".txt";
26             }
27             try {
28                 FileWriter writer = new FileWriter(filePath);
29                 BufferedWriter bw = new BufferedWriter(writer);
30                 bw.write(textArea.getText());
31                 bw.close();
32                 writer.close();
33                 currentFile = filePath;
34                 JOptionPane.showMessageDialog(this, message: "File saved successfully!", title: "Success", JOptionPane.INFORMATION_MESSAGE);
35             } catch (Exception ex) {
36                 ex.printStackTrace();
37                 JOptionPane.showMessageDialog(this, message: "Error saving file!", title: "Error", JOptionPane.ERROR_MESSAGE);
38             }
39         }
40     }
41 }
```

OUTPUT: -

