import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class ScientificCalculatorWithModes extends JFrame implements ActionListener {

private final JTextField display;

private final JTextArea historyArea;

private boolean isDarkMode = true;

private boolean isScientificMode = false; // To toggle between modes

private JPanel buttonPanel; // Panel to hold the buttons

private double memoryValue = 0;

private String operator = "";

private double num1 = 0;

public ScientificCalculatorWithModes() {

// Frame settings

setTitle("Scientific Calculator");

setSize(600, 800);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

// Display and history area

display = new JTextField();

display.setFont(new Font("Arial", Font.BOLD, 30));

display.setHorizontalAlignment(JTextField.RIGHT);

display.setEditable(false);

add(display, BorderLayout.NORTH);

historyArea = new JTextArea(5, 20);

historyArea.setFont(new Font("Arial", Font.PLAIN, 18));

historyArea.setEditable(false);

JScrollPane historyScroll = new JScrollPane(historyArea);

historyScroll.setBorder(BorderFactory.createTitledBorder("History"));

add(historyScroll, BorderLayout.EAST);

// Buttons panel

buttonPanel = new JPanel();

buttonPanel.setLayout(new GridLayout(6, 5, 5, 5));

add(buttonPanel, BorderLayout.CENTER);

// Add default buttons (basic functions)

addBasicButtons();

// Toggle mode buttons

JPanel modePanel = new JPanel(new GridLayout(1, 2));

JButton toggleTheme = new JButton("Light/Dark Mode");

toggleTheme.setFont(new Font("Arial", Font.BOLD, 18));

toggleTheme.addActionListener(e -> toggleTheme());

modePanel.add(toggleTheme);

JButton toggleScientificMode = new JButton("Scientific Mode");

toggleScientificMode.setFont(new Font("Arial", Font.BOLD, 18));

toggleScientificMode.addActionListener(e -> toggleScientificMode());

modePanel.add(toggleScientificMode);

add(modePanel, BorderLayout.SOUTH);

// Apply dark mode initially

applyDarkMode();

setVisible(true);

}

private void toggleTheme() {

isDarkMode = !isDarkMode;

if (isDarkMode) {

applyDarkMode();

} else {

applyLightMode();

}

}

private void toggleScientificMode() {

isScientificMode = !isScientificMode;

buttonPanel.removeAll(); // Clear all existing buttons

if (isScientificMode) {

addBasicButtons();

addScientificButtons();

} else {

addBasicButtons();

}

buttonPanel.revalidate();

buttonPanel.repaint();

}

private void addBasicButtons() {

String[] basicButtons = {

"7", "8", "9", "÷", "C",

"4", "5", "6", "×", "MC",

"1", "2", "3", "-", "MR",

"+/-", "0", ".", "+", "=",

"x²", "√", "xʸ", "10ˣ", "mod",

"log", "ln", "|x|", "exp", "n!"

};

for (String text : basicButtons) {

JButton button = new JButton(text);

button.setFont(new Font("Arial", Font.BOLD, 20));

button.addActionListener(this);

buttonPanel.add(button);

}

}

private void addScientificButtons() {

String[] trigButtons = {

"sin", "cos", "tan", "sec", "csc",

"cot", "π", "e", "(", ")"

};

for (String text : trigButtons) {

JButton button = new JButton(text);

button.setFont(new Font("Arial", Font.BOLD, 20));

button.addActionListener(this);

buttonPanel.add(button);

}

}

private void applyDarkMode() {

getContentPane().setBackground(Color.BLACK);

display.setBackground(Color.DARK\_GRAY);

display.setForeground(Color.WHITE);

historyArea.setBackground(Color.DARK\_GRAY);

historyArea.setForeground(Color.WHITE);

}

private void applyLightMode() {

getContentPane().setBackground(Color.WHITE);

display.setBackground(Color.LIGHT\_GRAY);

display.setForeground(Color.BLACK);

historyArea.setBackground(Color.LIGHT\_GRAY);

historyArea.setForeground(Color.BLACK);

}

@Override

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

try {

if ("0123456789.".contains(command)) {

display.setText(display.getText() + command);

} else if ("+-×÷".contains(command)) {

num1 = Double.parseDouble(display.getText());

operator = command;

display.setText("");

} else if ("=".equals(command)) {

double num2 = Double.parseDouble(display.getText());

double result = switch (operator) {

case "+" -> num1 + num2;

case "-" -> num1 - num2;

case "×" -> num1 \* num2;

case "÷" -> num2 != 0 ? num1 / num2 : Double.NaN;

case "mod" -> num1 % num2;

case "xʸ" -> Math.pow(num1, num2);

default -> 0;

};

historyArea.append(num1 + " " + operator + " " + num2 + " = " + result + "\n");

display.setText(String.valueOf(result));

operator = "";

} else if ("C".equals(command)) {

display.setText("");

num1 = 0;

operator = "";

} else if ("MC".equals(command)) {

memoryValue = 0;

} else if ("MR".equals(command)) {

display.setText(String.valueOf(memoryValue));

} else if ("M+".equals(command)) {

memoryValue += Double.parseDouble(display.getText());

} else if ("M-".equals(command)) {

memoryValue -= Double.parseDouble(display.getText());

} else if ("π".equals(command)) {

display.setText(String.valueOf(Math.PI));

} else if ("e".equals(command)) {

display.setText(String.valueOf(Math.E));

} else if ("x²".equals(command)) {

double value = Math.pow(Double.parseDouble(display.getText()), 2);

display.setText(String.valueOf(value));

} else if ("√".equals(command)) {

double value = Math.sqrt(Double.parseDouble(display.getText()));

display.setText(String.valueOf(value));

} else if ("10ˣ".equals(command)) {

double value = Math.pow(10, Double.parseDouble(display.getText()));

display.setText(String.valueOf(value));

} else if ("log".equals(command)) {

double value = Math.log10(Double.parseDouble(display.getText()));

display.setText(String.valueOf(value));

} else if ("ln".equals(command)) {

double value = Math.log(Double.parseDouble(display.getText()));

display.setText(String.valueOf(value));

} else if ("exp".equals(command)) {

double value = Math.exp(Double.parseDouble(display.getText()));

display.setText(String.valueOf(value));

} else if ("n!".equals(command)) {

int value = factorial(Integer.parseInt(display.getText()));

display.setText(String.valueOf(value));

} else if ("|x|".equals(command)) {

double value = Math.abs(Double.parseDouble(display.getText()));

display.setText(String.valueOf(value));

} else if ("sin".equals(command)) {

double value = Math.sin(Math.toRadians(Double.parseDouble(display.getText())));

display.setText(String.valueOf(value));

} else if ("cos".equals(command)) {

double value = Math.cos(Math.toRadians(Double.parseDouble(display.getText())));

display.setText(String.valueOf(value));

} else if ("tan".equals(command)) {

double value = Math.tan(Math.toRadians(Double.parseDouble(display.getText())));

display.setText(String.valueOf(value));

} else if ("sec".equals(command)) {

double value = 1 / Math.cos(Math.toRadians(Double.parseDouble(display.getText())));

display.setText(String.valueOf(value));

} else if ("csc".equals(command)) {

double value = 1 / Math.sin(Math.toRadians(Double.parseDouble(display.getText())));

display.setText(String.valueOf(value));

} else if ("cot".equals(command)) {

double value = 1 / Math.tan(Math.toRadians(Double.parseDouble(display.getText())));

display.setText(String.valueOf(value));

}

} catch (Exception ex) {

display.setText("Error");

}

}

private int factorial(int num) {

if (num == 0 || num == 1) return 1;

return num \* factorial(num - 1);

}

public static void main(String[] args) {

new ScientificCalculatorWithModes();

}

}