```java
package expenseandincome_tracker_with_database;


import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridLayout;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.SwingConstants;
import javax.swing.border.Border;
import javax.swing.border.LineBorder;
import javax.swing.plaf.basic.BasicScrollBarUI;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableCellRenderer;



import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableCellRenderer;


/**
 *
 * @author 1BestCsharp
 */
public class ExpenseAndIncomeTrackerApp {

  // Variables for the main frame and UI components
  private JFrame frame;
  private JPanel titleBar;
  private JLabel titleLabel;
  private JLabel closeLabel;
  private JLabel minimizeLabel;
  private JPanel dashboardPanel;
  private JPanel buttonsPanel;
  private JButton addTransactionButton;
  private JButton removeTransactionButton;
  private JTable transactionTable;
  private DefaultTableModel tableModel;

  // Variable to store the total amount
  private double totalAmount = 0.0;

  // ArrayList to store data panel values
  private ArrayList<String> dataPanelValues = new ArrayList<>();

  // variables for form dragging
  private boolean isDragging = false;
  private Point mouseOffset;

  // Constructor
  public ExpenseAndIncomeTrackerApp(){
      frame = new JFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(800,500);
      frame.setLocationRelativeTo(null);
      // Remove form border and default close and minimize buttons
      frame.setUndecorated(true);
      // Set Custom border to the frame
      frame.getRootPane().setBorder(BorderFactory.createMatteBorder(5, 5, 5, 5, new Color(52, 73, 94)));

      // Create and set up the title bar
      titleBar = new JPanel();
      titleBar.setLayout(null);
      titleBar.setBackground(new Color(52,73,94));
      titleBar.setPreferredSize(new Dimension(frame.getWidth(), 30));
      frame.add(titleBar, BorderLayout.NORTH);

      // Create and set up the title label
      titleLabel = new JLabel("Expense And Income Tracker");
```

```
titleLabel.setForeground(Color.WHITE);
titleLabel.setFont(new Font("Arial", Font.BOLD, 17));
titleLabel.setBounds(10,0,250,30);
titleBar.add(titleLabel);

// Create and set up the close label
closeLabel = new JLabel("x");
closeLabel.setForeground(Color.WHITE);
closeLabel.setFont(new Font("Arial", Font.BOLD, 17));
closeLabel.setHorizontalAlignment(SwingConstants.CENTER);
closeLabel.setBounds(frame.getWidth() - 50, 0, 30, 30);
closeLabel.setCursor(new Cursor(Cursor.HAND_CURSOR));

// Add mouse listeners for close label interactions
closeLabel.addMouseListener(new MouseAdapter() {

    @Override
    public void mouseClicked(MouseEvent e){
        System.exit(0);
    }

    @Override
    public void mouseEntered(MouseEvent e){
        closeLabel.setForeground(Color.red);
    }

    @Override
    public void mouseExited(MouseEvent e){
        closeLabel.setForeground(Color.white);
    }

});
```

```
titleBar.add(closeLabel);

// Create and set up the minimize label
minimizeLabel = new JLabel("-");
minimizeLabel.setForeground(Color.WHITE);
minimizeLabel.setFont(new Font("Arial", Font.BOLD, 17));
minimizeLabel.setHorizontalAlignment(SwingConstants.CENTER);
minimizeLabel.setBounds(frame.getWidth() - 80, 0, 30, 30);
minimizeLabel.setCursor(new Cursor(Cursor.HAND_CURSOR));

// Add mouse listeners for minimize label interactions
minimizeLabel.addMouseListener(new MouseAdapter() {

   @Override
   public void mouseClicked(MouseEvent e){
      frame.setState(JFrame.ICONIFIED);
   }

   @Override
   public void mouseEntered(MouseEvent e){
      minimizeLabel.setForeground(Color.red);
   }

   @Override
   public void mouseExited(MouseEvent e){
      minimizeLabel.setForeground(Color.white);
   }

});

titleBar.add(minimizeLabel);

// Set up form dragging functionality
```

```java
// Mouse listener for window dragging
titleBar.addMouseListener(new MouseAdapter() {

    @Override
    public void mousePressed(MouseEvent e){
        isDragging = true;
        mouseOffset = e.getPoint();
    }

    @Override
    public void mouseReleased(MouseEvent e){
        isDragging = false;
    }

});

// Mouse motion listener for window dragging
titleBar.addMouseMotionListener(new MouseAdapter() {

    @Override
    public void mouseDragged(MouseEvent e){
        if(isDragging){
            // When the mouse is dragged, this event is triggered
            // Get the current location of the mouse on the screen
            Point newLocation = e.getLocationOnScreen();
            // Calculate the new window location by adjusting for the initial mouse offset
            newLocation.translate(-mouseOffset.x, -mouseOffset.y);
            // Set the new location of the main window to achieve dragging effect
            frame.setLocation(newLocation);
        }
    }

});
```

```
// Create and set up the dashboard panel
dashboardPanel = new JPanel();
dashboardPanel.setLayout(new FlowLayout(FlowLayout.CENTER,20,20));
dashboardPanel.setBackground(new Color(236,240,241));
frame.add(dashboardPanel,BorderLayout.CENTER);

// Calculate total amount and populate data panel values
totalAmount = TransactionValuesCalculation.getTotalValue(TransactionDAO.getAllTransaction());
dataPanelValues.add(String.format("-$%,.2f", TransactionValuesCalculation.getTotalExpenses(TransactionDAO.getAllTransaction())));
dataPanelValues.add(String.format("$%,.2f", TransactionValuesCalculation.getTotalIncomes(TransactionDAO.getAllTransaction())));
dataPanelValues.add("$"+totalAmount);



// Add data panels for Expense, Income, and Total
addDataPanel("Expense", 0);
addDataPanel("Income", 1);
addDataPanel("Total", 2);



// Create and set up buttons panel
addTransactionButton = new JButton("Add Transaction");
addTransactionButton.setBackground(new Color(41,128,185));
addTransactionButton.setForeground(Color.WHITE);
addTransactionButton.setFocusPainted(false);
addTransactionButton.setBorderPainted(false);
addTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
addTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
addTransactionButton.addActionListener((e) -> { showAddTransactionDialog(); });
```

```
removeTransactionButton = new JButton("Remove Transaction");
removeTransactionButton.setBackground(new Color(231,76,60));
removeTransactionButton.setForeground(Color.WHITE);
removeTransactionButton.setFocusPainted(false);
removeTransactionButton.setBorderPainted(false);
removeTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
removeTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
removeTransactionButton.addActionListener((e) -> {
    removeSelectedTransaction();
});

buttonsPanel = new JPanel();
buttonsPanel.setLayout(new BorderLayout(10, 5));
buttonsPanel.add(addTransactionButton, BorderLayout.NORTH);
buttonsPanel.add(removeTransactionButton, BorderLayout.SOUTH);
dashboardPanel.add(buttonsPanel);

// Set up the transaction table
String[] columnNames = {"ID","Type","Description","Amount"};
tableModel = new DefaultTableModel(columnNames, 0){
    @Override
    public boolean isCellEditable(int row, int column){
        // Make all cells non-editable
        return false;
    }

};


transactionTable = new JTable(tableModel);
configureTransactionTable();

JScrollPane scrollPane = new JScrollPane(transactionTable);
```

```
    configureScrollPane(scrollPane);
    dashboardPanel.add(scrollPane);

    frame.setVisible(true);
}


// fix the negative value
private String fixNegativeValueDisplay(double value){

    // Check if the input starts with "$-" (indicating negative)
    String newVal = String.format("$%.2f", value);

    if(newVal.startsWith("$-")){
        // Extract the numeric part after "$-"
        String numericPart = newVal.substring(2);
        // Format the result as "-$XXX"
        newVal = "-$"+numericPart;
    }

    return newVal;
}


// Removes the selected transaction from the table and database
private void removeSelectedTransaction(){

    int selectedRow = transactionTable.getSelectedRow();

    // Check if a row is selected
    if(selectedRow != -1){
        // Obtain the transaction details from the selected row
        int transactionId = (int) transactionTable.getValueAt(selectedRow, 0);
```

```
String type = transactionTable.getValueAt(selectedRow, 1).toString();
String amountStr = transactionTable.getValueAt(selectedRow, 3).toString();
double amount = Double.parseDouble(amountStr.replace("$", "").replace(" ", "").replace(",", ""));

// Update totalAmount based on the type of transaction
if(type.equals("Income")){ totalAmount -= amount; }
else{ totalAmount += amount; }

// Repaint the total panel to reflect the updated total amount
JPanel totalPanel = (JPanel) dashboardPanel.getComponent(2);
totalPanel.repaint();

// Determine the index of the data panel to update (0 for Expense, 1 for Income)
int indexToUpdate = type.equals("Income") ? 1 : 0;

// Update the data panel value and repaint it
String currentValue = dataPanelValues.get(indexToUpdate);
double currentAmount = Double.parseDouble(currentValue.replace("$", "").replace(" ", "").replace(",", "").replace("--", "-"));
double updatedAmount = currentAmount + (type.equals("Income") ? -amount : amount);
//dataPanelValues.set(indexToUpdate, String.format("$%,.2f",updatedAmount));
if(indexToUpdate == 1){ // income
   dataPanelValues.set(indexToUpdate, String.format("$%,.2f", updatedAmount));
}
// expense
else{ dataPanelValues.set(indexToUpdate, fixNegativeValueDisplay(updatedAmount)); }


// Repaint the corresponding data panel
JPanel dataPanel = (JPanel) dashboardPanel.getComponent(indexToUpdate);
dataPanel.repaint();

// Remove the selected row from the table model
tableModel.removeRow(selectedRow);
```

```
        // Remove the transaction from the database
        removeTransactionFromDatabase(transactionId);
    }
}



// Remove a transaction from the database
private void removeTransactionFromDatabase(int transactionId){

    try {
        Connection connection = DatabaseConnection.getConnection();
        PreparedStatement ps = connection.prepareStatement("DELETE FROM transaction_table WHERE id = ?");

        ps.setInt(1, transactionId);
        ps.executeLargeUpdate();
        System.out.println("Transaction Removed");

    } catch (SQLException ex) {
        Logger.getLogger(ExpenseAndIncomeTrackerApp.class.getName()).log(Level.SEVERE, null, ex);
    }


}



// Displays the dialog for adding a new transaction
private void showAddTransactionDialog(){

    // Create a new JDialog for adding a transaction
    JDialog dialog = new JDialog(frame, "Add Transaction", true);
    dialog.setSize(400,250);
dialog.setLocationRelativeTo(frame);

// Create a panel to hold the components in a grid layout
JPanel dialogPanel = new JPanel(new GridLayout(4, 0, 10, 10));
// Set an empty border with padding for the dialog panel
dialogPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
dialogPanel.setBackground(Color.LIGHT_GRAY);

// Create and configure components for transaction input
JLabel typeLabel = new JLabel("Type:");
JComboBox<String> typeCombobox = new JComboBox<>(new String[]{"Expense","Income"});
typeCombobox.setBackground(Color.WHITE);
typeCombobox.setBorder(BorderFactory.createLineBorder(Color.yellow));

JLabel descriptionLabel = new JLabel("Description:");
JTextField descriptionField = new JTextField();
descriptionField.setBorder(BorderFactory.createLineBorder(Color.yellow));

JLabel amountLabel = new JLabel("Amount:");
JTextField amountField = new JTextField();
amountField.setBorder(BorderFactory.createLineBorder(Color.yellow));

// Create and configure the "Add" button
JButton addButton = new JButton("Add");
addButton.setBackground(new Color(41,128,185));
addButton.setForeground(Color.WHITE);
addButton.setFocusPainted(false);
addButton.setBorderPainted(false);
addButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
addButton.addActionListener((e) -> {
    addTransaction(typeCombobox, descriptionField, amountField);
});
```

```
    // Add components to the dialog panel
    dialogPanel.add(typeLabel);
    dialogPanel.add(typeCombobox);
    dialogPanel.add(descriptionLabel);
    dialogPanel.add(descriptionField);
    dialogPanel.add(amountLabel);
    dialogPanel.add(amountField);
    dialogPanel.add(new JLabel()); // Empty label for spacing
    dialogPanel.add(addButton);


    DatabaseConnection.getConnection();

    dialog.add(dialogPanel);
    dialog.setVisible(true);

}


// Add a new transaction to the database
private void addTransaction(JComboBox<String> typeCombobox, JTextField descriptionField, JTextField amountField){


        // Retrieve transaction details from the input fields
        String type = (String) typeCombobox.getSelectedItem();
        String description = descriptionField.getText();
        String amount = amountField.getText();


        // Parse the amount string to a double value
        double newAmount = Double.parseDouble(amount.replace("$", "").replace(" ", "").replace(",", ""));

        // Update the total amount based on the transaction type (Income or Expense)
// Income
if(type.equals("Income")){ totalAmount += newAmount; }
// Expense
else{ totalAmount -= newAmount; }

// Update the displayed total amount on the dashboard panel
JPanel totalPanel = (JPanel) dashboardPanel.getComponent(2);
totalPanel.repaint();

// Determine the index of the data panel to update based on the transaction type
int indexToUpdate = type.equals("Income") ? 1 : 0;

// Retrieve the current value of the data panel
String currentValue = dataPanelValues.get(indexToUpdate);

// Parse the current amount string to a double value
double currentAmount = Double.parseDouble(currentValue.replace("$", "").replace(" ", "").replace(",", ""));

// Calculate the updated amount based on the transaction type
double updatedAmount = currentAmount + (type.equals("Income") ? newAmount : -newAmount);

// Update the data panel with the new amount
if(indexToUpdate == 1){ // income
   dataPanelValues.set(indexToUpdate, String.format("$%,.2f", updatedAmount));
}
// expense
else{ dataPanelValues.set(indexToUpdate, fixNegativeValueDisplay(updatedAmount)); }

// Update the displayed data panel on the dashboard panel
JPanel dataPanel = (JPanel) dashboardPanel.getComponent(indexToUpdate);
dataPanel.repaint();
```

```
        try {
            Connection connection = DatabaseConnection.getConnection();
            String insertQuery = "INSERT INTO transaction_table(transaction_type, description, amount) VALUES (?,?,?)";
            PreparedStatement ps = connection.prepareStatement(insertQuery);


            ps.setString(1, type);
            ps.setString(2, description);
            ps.setDouble(3, Double.parseDouble(amount));
            ps.executeUpdate();
            System.out.println("Data inserted successfully.");

            tableModel.setRowCount(0);
            populateTableTransactions();


        } catch (SQLException ex) {
          System.out.println("Error - Data not inserted.");
        }





    }


    // Populate Table Transactions
    private void populateTableTransactions(){

      for(Transaction transaction : TransactionDAO.getAllTransaction()){
        Object[] rowData = { transaction.getId(), transaction.getType(),
                          transaction.getDescription(), transaction.getAmount() };
        tableModel.addRow(rowData);

      }

    }


    // Configures the appearance and behavior of the transaction table
    private void configureTransactionTable(){
      transactionTable.setBackground(new Color(236,240,241));
      transactionTable.setRowHeight(30);
      transactionTable.setShowGrid(false);
      transactionTable.setBorder(null);
      transactionTable.setFont(new Font("Arial",Font.ITALIC,16));
      transactionTable.setDefaultRenderer(Object.class, new TransactionTableCellRenderer());
      transactionTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

      populateTableTransactions();

      JTableHeader tableHeader = transactionTable.getTableHeader();
      tableHeader.setForeground(Color.red);
      tableHeader.setFont(new Font("Arial", Font.BOLD, 18));
      tableHeader.setDefaultRenderer(new GradientHeaderRenderer());
    }


    // Configures the appearance of the scroll pane
    private void configureScrollPane(JScrollPane scrollPane){

      scrollPane.getVerticalScrollBar().setUI(new CustomScrollBarUI());
      scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
      scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
      scrollPane.setPreferredSize(new Dimension(750, 300));
```

```
}


// Add a data panel to the dashboard panel
private void addDataPanel(String title, int index){
    // Create a new JPanel for the data panel
    JPanel dataPanel = new JPanel(){
        // Override the paintComponent method to customize the appearance
        @Override
        protected void paintComponent(Graphics g){
            // Call the paintComponent method of the superclass
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D) g;
            //make the drawing smooth
            g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
            // Check if the title is "Total" to determine the content to display
            if(title.equals("Total")){
                // If the title is "Total," draw the data panel with the total amount
                //drawDataPanel(g2d, title, String.format("$%,.2f", totalAmount), getWidth(), getHeight());
                drawDataPanel(g2d, title, fixNegativeValueDisplay(totalAmount), getWidth(), getHeight());
            }
            else{
                // If the title is not "Total," draw the data panel with the corresponding value from the list
                drawDataPanel(g2d, title, dataPanelValues.get(index), getWidth(), getHeight());
            }
        }

    };

    // Set the layout, size, background color, and border for the data panel
    dataPanel.setLayout(new GridLayout(2, 1));
    dataPanel.setPreferredSize(new Dimension(170, 100));
    dataPanel.setBackground(new Color(255,255,255));
    dataPanel.setBorder(new LineBorder(new Color(149,165,166),2));
    dashboardPanel.add(dataPanel);
}



// Draws a data panel with specified title and value
private void drawDataPanel(Graphics g, String title, String value, int width, int height){

    Graphics2D g2d = (Graphics2D)g;

    // draw the panel
    g2d.setColor(new Color(255,255,255));
    g2d.fillRoundRect(0, 0, width, height, 20, 20);
    g2d.setColor(new Color(236,240,241));
    g2d.fillRect(0, 0, width, 40);

    // draw title
    g2d.setColor(Color.BLACK);
    g2d.setFont(new Font("Arial", Font.BOLD, 20));
    g2d.drawString(title, 20, 30);

    // draw value
    g2d.setColor(Color.BLACK);
    g2d.setFont(new Font("Arial", Font.PLAIN, 16));
    g2d.drawString(value, 20, 75);


}

// main method
public static void main(String[] args) {
        new ExpenseAndIncomeTrackerApp();
    }


}





// Custom table header renderer with gradient background
class GradientHeaderRenderer extends JLabel implements TableCellRenderer{

    private final Color startColor = new Color(192,192,192);
    private final Color endColor = new Color(50,50,50);


    public GradientHeaderRenderer(){
        setOpaque(false);
        setHorizontalAlignment(SwingConstants.CENTER);
        setForeground(Color.WHITE);
        setFont(new Font("Arial", Font.BOLD,22));
        setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createMatteBorder(0, 0, 1, 1, Color.YELLOW),
                BorderFactory.createEmptyBorder(2, 5, 2, 5))
        );
    }

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {

        setText(value.toString());
        return this;
```

```
  }

  @Override
  protected void paintComponent(Graphics g){

      Graphics2D g2d = (Graphics2D) g;

      int width = getWidth();
      int height = getHeight();

      GradientPaint gradientPaint = new GradientPaint(
              0, 0, startColor,width, 0, endColor);

      g2d.setPaint(gradientPaint);
      g2d.fillRect(0, 0, width, height);

      super.paintComponent(g);
  }

}




// Create a custom scroll bar UI class for the scrollPane
class CustomScrollBarUI extends BasicScrollBarUI{
     // Colors for the thumb and track of the scroll bar
     private Color thumbColor = new Color(189,195,199);
     private Color trackColor = new Color(236,240,241);

     // Override method to configure the scroll bar colors
     @Override
```

```
protected void configureScrollBarColors(){
   // Call the superclass method to ensure default configuration
   super.configureScrollBarColors();

}

// Override method to create the decrease button of the scroll bar
@Override
protected JButton createDecreaseButton(int orientation){
   // Create an empty button for the decrease button
   return createEmptyButton();
}

// Override method to create the increase button of the scroll bar
@Override
protected JButton createIncreaseButton(int orientation){
   // Create an empty button for the increase button
   return createEmptyButton();
}

// Override method to paint the thumb of the scroll bar
@Override
protected void paintThumb(Graphics g, JComponent c, Rectangle thumbBounds){
   // Set the color and fill the thumb area with the specified color
   g.setColor(thumbColor);
   g.fillRect(thumbBounds.x, thumbBounds.y, thumbBounds.width, thumbBounds.height);
}

// Override method to paint the track of the scroll bar
@Override
protected void paintTrack(Graphics g, JComponent c, Rectangle trackBounds){
   // Set the color and fill the track area with the specified color
   g.setColor(trackColor);
```

```
            g.fillRect(trackBounds.x, trackBounds.y, trackBounds.width, trackBounds.height);
        }

        // Private method to create an empty button with zero dimensions
        private JButton createEmptyButton(){
            JButton button = new JButton();
            button.setPreferredSize(new Dimension(0, 0));
            button.setMaximumSize(new Dimension(0, 0));
            button.setMinimumSize(new Dimension(0, 0));
            return button;
        }

}

// Custom cell renderer for the transaction table
class TransactionTableCellRenderer extends DefaultTableCellRenderer{

    // Override method to customize the rendering of table cells
    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column){

        // Call the superclass method to get the default rendering component
        Component c = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
        // Get the transaction type from the second column of the table
        String type = (String) table.getValueAt(row, 1);

        // Customize the appearance based on the selection and transaction type
        if(isSelected){
            c.setForeground(Color.BLACK);
            c.setBackground(Color.ORANGE);
        }
        else
        {
            if("Income".equals(type)){
                c.setBackground(new Color(144, 238, 144));
            }
            else{
                c.setBackground(new Color(255,99,71));
            }
        }

        return c;
    }
}
```