```
package ia;

import java.util.Scanner;
import java.io.*;
import java.awt.*;
import javax.swing.*;

public class Main{
	static LinkedList i = new LinkedList();
	static LinkedList exp = new LinkedList();
	static Font ft;
	static ImageIcon icon = new ImageIcon("./Images/sc.png");
	public static void main(String args[]) {

		// Font
		try {
			ft = Font.createFont(Font.TRUETYPE_FONT, new
File("./Fonts/Futura.ttf")).deriveFont(20f);
			GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
			ge.registerFont(ft);
		} catch (IOException e) {
			e.printStackTrace();
		} catch(FontFormatException e) {
			e.printStackTrace();
		}

		// Login Screen
		LoginScreen ls = new LoginScreen();
		ls.setVisible(true);
	  ls.setLocationRelativeTo(null);
		ls.setResizable(false);
    }

	// Create the LinkedLists of Income/Expenditure
	public static void initLists(){
		File f = new File("./TextFiles/" + LoginScreen.currentUser +
"info.txt");
	    File fe = new File("./TextFiles/" + LoginScreen.currentUser +
"infoExp.txt");
		// Reading existing income data
		try {
			Scanner sc = new Scanner(f);
			while (sc.hasNextLine()) {
				String data = sc.nextLine();
				String month = data.substring(0, data.indexOf(' '));
				double taxRate =
Double.parseDouble(data.substring(data.indexOf(' ')+1, data.indexOf('-')));
				double rawIncome =
Double.parseDouble(data.substring(data.indexOf('-')+1, data.length()));
				i.insert(Income.totalMonths, new Income(month, taxRate,
rawIncome));
			}
			sc.close();
		} catch (FileNotFoundException e) {
			System.out.println("An error occurred.");
			e.printStackTrace();
		}
```

```
			// Reading existing expenditure data
			try {
				Scanner sc = new Scanner(fe);
				while (sc.hasNextLine()) {
					String data = sc.nextLine();
					String month = data.substring(0, data.indexOf(' '));
					double lux =
Double.parseDouble(data.substring(data.indexOf('l')+1, data.indexOf('f')));
					double food =
Double.parseDouble(data.substring(data.indexOf('f')+1, data.indexOf('t')));
					double transp =
Double.parseDouble(data.substring(data.indexOf('t')+1, data.indexOf('u')));
					double utility =
Double.parseDouble(data.substring(data.indexOf('u')+1, data.indexOf('o')));
					double other =
Double.parseDouble(data.substring(data.indexOf('o')+1, data.length()));
					exp.insert(Expenditure.totalEntries, new
Expenditure(month, lux, food, transp, utility, other));
				}
				sc.close();
			} catch (FileNotFoundException e) {
				System.out.println("An error occurred.");
				e.printStackTrace();
			}
		}
}

package ia;
import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ChangePastData extends JFrame {
    private JButton jButton1;
    private JButton jButton2;
    private JComboBox<String> jComboBox1;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JOptionPane jOptionPane1;
    private JTextField jTextField1;
    private ImagePanel cpdPanel = new ImagePanel(new
ImageIcon("./Images/cpd.png").getImage());

    public ChangePastData() {
        initComponents();
    }

    private void initComponents() {

        jComboBox1 = new JComboBox<>();
        jTextField1 = new JTextField();
        jButton1 = new JButton();
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();
        jLabel3 = new JLabel();
        jButton2 = new JButton();
```

```
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jComboBox1.setModel(new DefaultComboBoxModel<>(new String[] { "January",
"February", "March", "April", "May", "June", "July", "August", "September",
"October", "December" }));

        jButton1.setText("Select");
        jButton1.setBackground(new Color(102,178,255));
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel1.setText("Change Selected Month - This allows you to view
personalized advice from past months as well ");
        jLabel2.setText("as alter data from that month. The menu screen will
display data from the selected month, and");
        jLabel3.setText("you may use the update buttons to change the data.");

        jButton2.setText("Exit");
        jButton2.setBackground(new Color(102,178,255));
        jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel2)
                            .addComponent(jLabel3)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jComboBox1,
GroupLayout.PREFERRED_SIZE, 302, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jTextField1,
GroupLayout.PREFERRED_SIZE, 79, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jButton1, GroupLayout.PREFERRED_SIZE,
82, GroupLayout.PREFERRED_SIZE)))
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(jButton2)))
```

```
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(49, 49, 49)
                .addComponent(jLabel1)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel3)
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jComboBox1, GroupLayout.PREFERRED_SIZE, 29,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE, 29,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton1, GroupLayout.PREFERRED_SIZE, 29,
GroupLayout.PREFERRED_SIZE))
                .addGap(67, 67, 67)
                .addComponent(jButton2)
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        this.add(cpdPanel);
        this.setIconImage(Main.icon.getImage());
             this.setTitle("SmartCash - Change Past Data");
        pack();
    }

    // Add Month Button
    private void jButton1ActionPerformed(ActionEvent evt) {
       String month = jComboBox1.getSelectedItem() + "";
        month = monthToNum(month);

             if(jTextField1.getText().equals("")){
                    jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"Please enter the year.");
             }
             else{
                    int year = Integer.parseInt(jTextField1.getText());
                    boolean valid = false;
                    for(int i = 0; i < Main.exp.size; i++) {
                           String mData = ((Expenditure)
Main.exp.lookUp(i)).getMonth().substring(0,2);
                           int yData = Integer.parseInt(((Expenditure)
Main.exp.lookUp(i)).getMonth().substring(2));
                           if(year == yData) {
                                  if(Integer.parseInt(month) ==
Integer.parseInt(mData)) {
                                         Income.index = i;
                                         Expenditure.index = i;
                                         valid = true;
                                         break;
                                  }
                           }
                    }
```

```
                    if(!valid) {
                           jOptionPane1 = new JOptionPane();
                           JFrame f = new JFrame();
                           JOptionPane.showMessageDialog(f,"Data for that month
does not exist.");
                    }
                    else {
                           Income.currentMonth = month + year;
                           Expenditure.currentMonth = month + year;
                           Menu.reload();
                           jOptionPane1 = new JOptionPane();
                           JFrame f = new JFrame();
                           JOptionPane.showMessageDialog(f,"You have successfully
selected a different month.");
                           this.setVisible(false);
                    }
             }
    }

    // Exit Button
    private void jButton2ActionPerformed(ActionEvent evt) {
       this.setVisible(false);
    }

    // Convert Month String (Which Includes Year) To Only Month
       private String monthToNum(String s){
             String monthNum = "";

             if(s.equals("January")){
                    monthNum += "01";
             }
             else if(s.equals("February")){
                    monthNum += "02";
             }
             else if(s.equals("March")){
                    monthNum += "03";
             }
             else if(s.equals("April")){
                    monthNum += "04";
             }
             else if(s.equals("May")){
                    monthNum += "05";
             }
             else if(s.equals("June")){
                    monthNum += "06";
             }
             else if(s.equals("July")){
                    monthNum += "07";
             }
             else if(s.equals("August")){
                    monthNum += "08";
             }
             else if(s.equals("September")){
                    monthNum += "09";
             }
             else if(s.equals("October")){
                    monthNum += "10";
             }
             else if(s.equals("November")){
```

```
                monthNum += "11";
            }
            else if(s.equals("December")){
                monthNum += "12";
            }
            return monthNum;
        }
}

package ia;

import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Expenditure extends Money{
        private double luxuryExp = 0;
        private double foodExp = 0;
        private double transportExp = 0;
        private double utilityExp = 0;
        private double otherExp = 0;
        static File f = new File("src/TextFiles/" + LoginScreen.currentUser +
"infoExp.txt");
        static String currentMonth = "000000";
        static int totalEntries = 0;
        static String expInfo = "";
        static int index = totalEntries-1;

        public Expenditure(String month, double luxuryExp, double foodExp, double
transportExp, double utilityExp, double otherExp){
            super(month);
            this.luxuryExp = luxuryExp;
            this.foodExp = foodExp;
            this.transportExp = transportExp;
            this.utilityExp = utilityExp;
            this.otherExp = otherExp;
            if(currentMonth.equals("000000")) {
                currentMonth = month;
            }
            try {
                Scanner sc = new Scanner(f);
                while (sc.hasNextLine()) {
                    String data = sc.nextLine();
                    if(!sc.hasNextLine()){
                        currentMonth = data.substring(0, data.indexOf('
'));
                    }
                    if(totalEntries == 0) {
                        expInfo += data + "\n";
                    }
                }
                sc.close();
            } catch (FileNotFoundException e) {
                System.out.println("An error occurred.");
                e.printStackTrace();
            }
            totalEntries++;
```

```
	}

	// Getter Methods
	public double getLuxuryExp(){
		return luxuryExp;
	}
	public double getFoodExp(){
		return foodExp;
	}
	public double getTransportExp(){
		return transportExp;
	}
	public double getUtilityExp(){
		return utilityExp;
	}
	public double getOtherExp(){
		return otherExp;
	}

	// Setter Methods
	public void setLuxuryExp(double a){
		luxuryExp = a;
	}
	public void setFoodExp(double a){
		foodExp = a;
	}
	public void setTransportExp(double a){
		transportExp = a;
	}
	public void setUtilityExp(double a){
		utilityExp = a;
	}
	public void setOtherExp(double a){
		otherExp = a;
	}

	// Calculate Expenditure of Most Recent Entry in File
	public static double calcExp(){
		try {
			Scanner sc = new Scanner(f);
			while (sc.hasNextLine()) {
				String data = sc.nextLine();
				if(data.substring(0, data.indexOf('
')).equals(currentMonth)){
					double lux =
Double.parseDouble(data.substring(data.indexOf('l')+1, data.indexOf('f')));
					double food =
Double.parseDouble(data.substring(data.indexOf('f')+1, data.indexOf('t')));
					double transp =
Double.parseDouble(data.substring(data.indexOf('t')+1, data.indexOf('u')));
					double utility =
Double.parseDouble(data.substring(data.indexOf('u')+1, data.indexOf('o')));
					double other =
Double.parseDouble(data.substring(data.indexOf('o')+1, data.length()));
					return lux+food+transp+utility+other;
				}

			}
			sc.close();
```

```
		} catch (FileNotFoundException e) {
			System.out.println("An error occurred.");
			e.printStackTrace();
		}
		return -1.0;
	}

	// Calculate Total Expenditure of Any Expenditure Item
	public double calculate() {
		return luxuryExp+foodExp+transportExp+utilityExp+otherExp;
	}

	// Update Input String
	public static void updateAll(){
		Expenditure.expInfo =
Expenditure.expInfo.substring(0,Expenditure.expInfo.lastIndexOf(' ')) + " " + "l" +
((Expenditure) Main.exp.lookUp(Expenditure.totalEntries-1)).getLuxuryExp() + "f" +
((Expenditure) Main.exp.lookUp(Expenditure.totalEntries-1)).getFoodExp() + "t" +
((Expenditure) Main.exp.lookUp(Expenditure.totalEntries-1)).getTransportExp() + "u"
+ ((Expenditure) Main.exp.lookUp(Expenditure.totalEntries-1)).getUtilityExp() + "o"
+ ((Expenditure) Main.exp.lookUp(Expenditure.totalEntries-1)).getOtherExp() + "\n";
	}
}

package ia;

import java.util.Scanner;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ExpUpdater extends JFrame {
    private JButton jButton1;
    private JButton jButton2;
    private JButton jButton3;
    private JButton jButton4;
    private JButton jButton5;
    private JButton jButton6;
    private JButton jButton7;
    private JComboBox<String> jComboBox1;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JLabel jLabel4;
    private JLabel jLabel5;
    private JLabel jLabel6;
    private JLabel jLabel7;
    private JLabel jLabel8;
    private JTextField jTextField1;
    private JTextField jTextField2;
    private JTextField jTextField3;
    private JTextField jTextField4;
    private JTextField jTextField5;
    private JTextField jTextField6;
      private JOptionPane jOptionPane1;
      private ImagePanel euPanel = new ImagePanel(new
ImageIcon("./Images/eu.png").getImage());
```

```
    public ExpUpdater() {
        initComponents();
    }

    private void initComponents() {

        jComboBox1 = new JComboBox<>();
        jTextField1 = new JTextField();
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();
        jButton1 = new JButton();
        jLabel3 = new JLabel();
        jLabel4 = new JLabel();
        jLabel5 = new JLabel();
        jLabel6 = new JLabel();
        jLabel7 = new JLabel();
        jLabel8 = new JLabel();
        jTextField2 = new JTextField();
        jButton2 = new JButton();
        jTextField3 = new JTextField();
        jTextField4 = new JTextField();
        jTextField5 = new JTextField();
        jTextField6 = new JTextField();
        jButton3 = new JButton();
        jButton4 = new JButton();
        jButton5 = new JButton();
        jButton6 = new JButton();
        jButton7 = new JButton();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jComboBox1.setModel(new DefaultComboBoxModel<>(new String[] { "Utility
Bills", "Food", "Luxury", "Transport", "Other" }));

        jLabel1.setText("Expenditure Amount");
        jLabel2.setText("Expenditure Type");

        jButton1.setText("Add Expenditure");
             jButton1.setBackground(new Color(102,178,255));
             jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel3.setText("Change Existing Expenditures:");
        jLabel4.setText("Utility Bills");
        jLabel5.setText("Food");
        jLabel6.setText("Luxury");
        jLabel7.setText("Transport");
        jLabel8.setText("Other");

        jButton2.setText("Update");
             jButton2.setBackground(new Color(102,178,255));
             jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
```

```
        jButton3.setText("Update");
             jButton3.setBackground(new Color(102,178,255));
             jButton3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jButton4.setText("Update");
             jButton4.setBackground(new Color(102,178,255));
             jButton4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });
        jButton5.setText("Update");
             jButton5.setBackground(new Color(102,178,255));
             jButton5.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });
        jButton6.setText("Update");
             jButton6.setBackground(new Color(102,178,255));
             jButton6.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
        });
        jButton7.setText("Exit");
             jButton7.setBackground(new Color(102,178,255));
        jButton7.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });

             if(Expenditure.totalEntries != 0){
                    for(int i = 0; i < Main.exp.size; i++) {
                           String mData = ((Expenditure)
Main.exp.lookUp(i)).getMonth();
                           if(Expenditure.currentMonth.equals(mData)) {
                                  jTextField2.setText(((Expenditure)
Main.exp.lookUp(i)).getUtilityExp()+"");
                                  jTextField3.setText(((Expenditure)
Main.exp.lookUp(i)).getFoodExp()+"");
                                  jTextField4.setText(((Expenditure)
Main.exp.lookUp(i)).getLuxuryExp()+"");
                                  jTextField5.setText(((Expenditure)
Main.exp.lookUp(i)).getTransportExp()+"");
                                  jTextField6.setText(((Expenditure)
Main.exp.lookUp(i)).getOtherExp()+"");
                                  Expenditure.index = i;
                                  break;
                           }
                    }
             }
```

```
        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(31, 31, 31)
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1)
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jButton7))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel3)
                            .addComponent(jLabel2)
                            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
                                    .addComponent(jComboBox1, 0, 160,
Short.MAX_VALUE)
                                    .addComponent(jTextField1))
                                .addGap(18, 18, 18)
                                .addComponent(jButton1)))
                        .addGap(0, 68, Short.MAX_VALUE))
                    .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel5)
                            .addComponent(jLabel6)
                            .addComponent(jLabel7)
                            .addComponent(jLabel8)
                            .addComponent(jLabel4))
                        .addGap(23, 23, 23)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jTextField2)
                            .addComponent(jTextField6,
GroupLayout.Alignment.TRAILING)
                            .addComponent(jTextField5)
                            .addComponent(jTextField4)
                            .addComponent(jTextField3))
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
                            .addComponent(jButton2, GroupLayout.DEFAULT_SIZE, 91,
Short.MAX_VALUE)
                            .addComponent(jButton3, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jButton4, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jButton5, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jButton6, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                .addContainerGap())
        );
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(44, 44, 44)
                        .addComponent(jLabel1)
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED))
                    .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jButton7)
                        .addGap(19, 19, 19)))

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton1))
                .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel2)
                .addGap(2, 2, 2)
                .addComponent(jComboBox1, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                .addGap(28, 28, 28)
                .addComponent(jLabel3)
                .addGap(9, 9, 9)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton2))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(jTextField3, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton3))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel6)
                    .addComponent(jTextField4, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton4))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel7)
                    .addComponent(jTextField5, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton5))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel8)
                    .addComponent(jTextField6, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(jButton6))
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

             this.add(euPanel);
             this.setIconImage(Main.icon.getImage());
             this.setTitle("SmartCash - Update Expenditure Data");
        pack();
    }

    // Add Expenditure Method
    private void jButton1ActionPerformed(ActionEvent evt) {
        int expType = jComboBox1.getSelectedIndex();
           try{
                  double expAmt = Double.parseDouble(jTextField1.getText());
                  if(expType == 0){
                        ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setUtilityExp(expAmt + ((Expenditure)
Main.exp.lookUp(Expenditure.index)).getUtilityExp());
                  }
                  else if(expType == 1){
                        ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setFoodExp(expAmt + ((Expenditure)
Main.exp.lookUp(Expenditure.index)).getFoodExp());
                  }
                  else if(expType == 2){
                        ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setLuxuryExp(expAmt + ((Expenditure)
Main.exp.lookUp(Expenditure.index)).getLuxuryExp());
                  }
                  else if(expType == 3){
                        ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setTransportExp(expAmt + ((Expenditure)
Main.exp.lookUp(Expenditure.index)).getTransportExp());
                  }
                  else if(expType == 4){
                        ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setOtherExp(expAmt + ((Expenditure)
Main.exp.lookUp(Expenditure.index)).getOtherExp());
                  }

                  if(Expenditure.index == Expenditure.totalEntries -1){
                     Expenditure.updateAll();
                     rewrite();
                  }
                  else {
                     rewriteFromScratch();
                  }

                     jOptionPane1 = new JOptionPane();
                     JFrame f = new JFrame();
                     JOptionPane.showMessageDialog(f,"You have successfully added an
expenditure.");
                     jTextField1.setText("");
                  Menu.reload();
                  this.reload();
           }
           catch(NumberFormatException e){
                 jTextField1.setText("Invalid value. Please try again.");
```

```
            }
        }

        // Update Specific Expenditure Methods
           private void jButton2ActionPerformed(ActionEvent evt) {
            try{
                      double expAmt = Double.parseDouble(jTextField2.getText());
                         ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setUtilityExp(expAmt);
                      if(Expenditure.index == Expenditure.totalEntries -1){
                      Expenditure.updateAll();
                      rewrite();
                     }
                     else {
                      rewriteFromScratch();
                     }
                      jOptionPane1 = new JOptionPane();
                      JFrame f = new JFrame();
                      JOptionPane.showMessageDialog(f,"You have successfully updated
your Utility expenditure data.");
                      jTextField1.setText("");
                    Menu.reload();
                    this.reload();
              }
              catch(NumberFormatException e){
                    jTextField2.setText("Invalid value. Please try again.");
              }
        }
           private void jButton3ActionPerformed(ActionEvent evt) {
            try{
                      double expAmt = Double.parseDouble(jTextField3.getText());
                         ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setFoodExp(expAmt);
                      if(Expenditure.index == Expenditure.totalEntries -1){
                      Expenditure.updateAll();
                      rewrite();
                     }
                     else {
                      rewriteFromScratch();
                     }
                      jOptionPane1 = new JOptionPane();
                      JFrame f = new JFrame();
                      JOptionPane.showMessageDialog(f,"You have successfully updated
your Food expenditure data.");
                      jTextField1.setText("");
                    Menu.reload();
                    this.reload();
              }
              catch(NumberFormatException e){
                    jTextField2.setText("Invalid value. Please try again.");
              }
        }
           private void jButton4ActionPerformed(ActionEvent evt) {
                  try{
                      double expAmt = Double.parseDouble(jTextField4.getText());
                         ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setLuxuryExp(expAmt);
                      if(Expenditure.index == Expenditure.totalEntries -1){
                      Expenditure.updateAll();
```

```
                    rewrite();
                }
                else {
                    rewriteFromScratch();
                }
                    jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"You have successfully updated
your Luxury expenditure data.");
                    jTextField1.setText("");
                Menu.reload();
                this.reload();
            }
            catch(NumberFormatException e){
                  jTextField2.setText("Invalid value. Please try again.");
            }
    }
       private void jButton5ActionPerformed(ActionEvent evt) {
        try{
                 double expAmt = Double.parseDouble(jTextField5.getText());
                    ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setTransportExp(expAmt);
                    if(Expenditure.index == Expenditure.totalEntries -1){
                    Expenditure.updateAll();
                    rewrite();
                }
                else {
                    rewriteFromScratch();
                }
                    jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"You have successfully updated
your Transport expenditure data.");
                    jTextField1.setText("");
                Menu.reload();
                this.reload();
            }
            catch(NumberFormatException e){
                  jTextField2.setText("Invalid value. Please try again.");
            }
    }

       private void jButton6ActionPerformed(ActionEvent evt) {
              try{
                 double expAmt = Double.parseDouble(jTextField6.getText());
                    ((Expenditure)
Main.exp.lookUp(Expenditure.index)).setOtherExp(expAmt);
                    if(Expenditure.index == Expenditure.totalEntries -1){
                    Expenditure.updateAll();
                    rewrite();
                }
                else {
                    rewriteFromScratch();
                }
                    jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"You have successfully updated
your Other expenditure data.");
                    jTextField1.setText("");
```

```
                Menu.reload();
                this.reload();
            }
            catch(NumberFormatException e){
                  jTextField2.setText("Invalid value. Please try again.");
            }
    }

       // Exit Button
    private void jButton7ActionPerformed(ActionEvent evt) {
             this.setVisible(false);
    }

    // Reload Menu
       private void reload(){
             if(Expenditure.totalEntries > 0){
                   jTextField2.setText(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getUtilityExp()+"");
                   jTextField3.setText(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getFoodExp()+"");
                   jTextField4.setText(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getLuxuryExp()+"");
                   jTextField5.setText(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getTransportExp()+"");
                   jTextField6.setText(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getOtherExp()+"");
             }
       }

       // Rewrite File by Updating Input Strings
       private void rewrite(){
             try {
                   FileWriter fw = new FileWriter(Expenditure.f);
                   BufferedWriter br = new BufferedWriter(fw);
                   br.write(Expenditure.expInfo);
                   br.close();
             } catch (FileNotFoundException e) {
                   System.out.println("An error occurred.");
                   e.printStackTrace();
             } catch (IOException e) {
                   System.out.println("An error occurred.");
                   e.printStackTrace();
             }
       }

       // Rewrite File by Rewriting Input Strings
       private void rewriteFromScratch() {
             Expenditure.expInfo = "";
             Income.incomeInfo = "";
             for(int i = 0; i < Main.i.size; i++) {
                   Income.incomeInfo += ((Income) Main.i.lookUp(i)).getMonth() + "
" + ((Income) Main.i.lookUp(i)).getTaxRate() + "-" + ((Income)
Main.i.lookUp(i)).getRawIncome() + "\n";
             }
             for(int i = 0; i < Main.exp.size; i++) {
                   Expenditure.expInfo += ((Expenditure)
Main.exp.lookUp(i)).getMonth() + " " + "l" + ((Expenditure)
Main.exp.lookUp(i)).getLuxuryExp() + "f" + ((Expenditure)
Main.exp.lookUp(i)).getFoodExp() + "t" + ((Expenditure)
```

```
Main.exp.lookUp(i)).getTransportExp() + "u" + ((Expenditure)
Main.exp.lookUp(i)).getUtilityExp() + "o" + ((Expenditure)
Main.exp.lookUp(i)).getOtherExp() + "\n";
			}
			try {
				FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "infoExp.txt");
				BufferedWriter br = new BufferedWriter(fw);
				br.write(Expenditure.expInfo);
				br.close();
			} catch (FileNotFoundException e) {
				System.out.println("An error occurred.");
				e.printStackTrace();
			} catch (IOException e) {
				System.out.println("An error occurred.");
				e.printStackTrace();
			}
			try {
				FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "info.txt");
				BufferedWriter br = new BufferedWriter(fw);
				br.write(Income.incomeInfo);
				br.close();
			} catch (FileNotFoundException e) {
				System.out.println("An error occurred.");
				e.printStackTrace();
			} catch (IOException e) {
				System.out.println("An error occurred.");
				e.printStackTrace();
			}
		}
}

package ia;

import java.util.Scanner;
import java.util.ArrayList;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HistoryViewer extends JFrame {
	private JButton jButton1;
    private JButton jButton3;
    private JButton jButton4;
    private JComboBox<String> jComboBox1;
    private JComboBox<String> jComboBox2;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JScrollPane jScrollPane1;
    private JTextArea jTextArea1;
    private JTextField jTextField1;
    private JTextField jTextField2;
    private JTextField jTextField3;
    private JTextField jTextField4;
    private JOptionPane jOptionPane1;
```

```
      private ImagePanel hvPanel = new ImagePanel(new
ImageIcon("./Images/hv.png").getImage());

    public HistoryViewer() {
        initComponents();
    }


    @SuppressWarnings("static-access")
       private void initComponents() {


        jComboBox1 = new JComboBox<>();
        jButton1 = new JButton();
        jComboBox2 = new JComboBox<>();
        jButton3 = new JButton();
        jLabel1 = new JLabel();
        jTextField1 = new JTextField();
        jLabel2 = new JLabel();
        jTextField2 = new JTextField();
        jLabel3 = new JLabel();
        jTextField3 = new JTextField();
        jScrollPane1 = new JScrollPane();
        jTextArea1 = new JTextArea();
        jButton4 = new JButton();
        jTextField4 = new JTextField();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jComboBox1.setModel(new DefaultComboBoxModel<>(new String[] { "January",
"February", "March", "April", "May", "June", "July", "August", "September",
"October", "December" }));
        jComboBox2.setModel(new DefaultComboBoxModel<>(new String[] { "Highest
Expenditure", "Lowest Expenditure", "Highest Income", "Lowest Income" }));

        jButton1.setText("Exit");
             jButton1.setBackground(new Color(102,178,255));
             jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton3.setText("Sort");
             jButton3.setBackground(new Color(102,178,255));
             jButton3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

             jButton4.setText("Search");
             jButton4.setBackground(new Color(102,178,255));
             jButton4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });
```

```
        jLabel1.setText("Income");
        jLabel2.setText("Total Expenditure");
        jLabel3.setText("Net Gain");
             jTextArea1.setEditable(false);
             jTextField1.setEditable(false);
             jTextField2.setEditable(false);
             jTextField3.setEditable(false);

        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);

        String text = "";
        for(int i = 0; i < Main.exp.size; i++) {
                    if(i != Main.exp.size-1) text+=Money.monthToString(((Money)
Main.exp.lookUp(i)).getMonth(), Income.totalMonths) + ", ";
                    else text+=Money.monthToString(((Money)
Main.exp.lookUp(i)).getMonth(), Income.totalMonths);
                }
        jTextArea1.setText(text);

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1)
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE, 357,
GroupLayout.PREFERRED_SIZE)
                        .addGap(44, 44, 44))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING, false)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel3)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jTextField3,
GroupLayout.PREFERRED_SIZE, 357, GroupLayout.PREFERRED_SIZE))
                            .addGroup(GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                                .addComponent(jLabel2)

.addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(jTextField2,
GroupLayout.PREFERRED_SIZE, 357, GroupLayout.PREFERRED_SIZE)))
                        .addGap(0, 43, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jComboBox1, GroupLayout.PREFERRED_SIZE, 297,
GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jTextField4)
```

```
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButton4, GroupLayout.PREFERRED_SIZE, 81,
GroupLayout.PREFERRED_SIZE))
                    .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(jButton1))
                    .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addComponent(jComboBox2, 0, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addGap(18, 18, 18)
                        .addComponent(jButton3, GroupLayout.PREFERRED_SIZE, 81,
GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(44, 44, 44)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jComboBox1, GroupLayout.PREFERRED_SIZE, 28,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton4, GroupLayout.PREFERRED_SIZE, 28,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField4, GroupLayout.PREFERRED_SIZE, 28,
GroupLayout.PREFERRED_SIZE))
                .addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE, 31,
GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE, 31,
GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField3, GroupLayout.PREFERRED_SIZE, 31,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))
                .addGap(31, 31, 31)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton3, GroupLayout.PREFERRED_SIZE, 28,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jComboBox2, GroupLayout.PREFERRED_SIZE, 28,
GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jScrollPane1, GroupLayout.PREFERRED_SIZE, 108,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 14,
Short.MAX_VALUE)
                .addComponent(jButton1)
```

```
                .addContainerGap())
        );

        this.add(hvPanel);
              this.setIconImage(Main.icon.getImage());
              this.setTitle("SmartCash - View Past Data");
        pack();
    }

    // Exit
       private void jButton1ActionPerformed(ActionEvent evt) {
              this.setVisible(false);
    }

       // Sort
       private void jButton3ActionPerformed(ActionEvent evt) {
              if(jComboBox2.getSelectedIndex() == 0) {
                     jTextArea1.setText(selecExpSortMax(Main.exp));
              }
              else if(jComboBox2.getSelectedIndex() == 1) {
                     jTextArea1.setText(selecExpSort(Main.exp));
              }
              else if(jComboBox2.getSelectedIndex() == 2) {
                     jTextArea1.setText(selecIncSortMax(Main.i));
              }
              else if(jComboBox2.getSelectedIndex() == 3) {
                     jTextArea1.setText(selecIncSort(Main.i));
              }
    }

       // Search
       private void jButton4ActionPerformed(ActionEvent evt) {
              String month = monthToNum(jComboBox1.getSelectedItem()+"") +
jTextField4.getText();
              int index = seqExpSearch(Main.exp, month);
              if(index == -1) {
                     jOptionPane1 = new JOptionPane();
                     JFrame f = new JFrame();
                     JOptionPane.showMessageDialog(f,"Month & year not found. Please
try again.");
              }
              else {
                     jTextField1.setText(((Income)
(Main.i.lookUp(index))).getIncome()+"");
                     jTextField2.setText(((Expenditure)
(Main.exp.lookUp(index))).calculate()+"");
                     jTextField3.setText(((Income)
(Main.i.lookUp(index))).getIncome() - ((Expenditure)
(Main.exp.lookUp(index))).calculate()+"");
              }
    }

       private String monthToNum(String s){
              String monthNum = "";

              if(s.equals("January")){
                     monthNum += "01";
              }
              else if(s.equals("February")){
```

```
			monthNum += "02";
		}
		else if(s.equals("March")){
			monthNum += "03";
		}
		else if(s.equals("April")){
			monthNum += "04";
		}
		else if(s.equals("May")){
			monthNum += "05";
		}
		else if(s.equals("June")){
			monthNum += "06";
		}
		else if(s.equals("July")){
			monthNum += "07";
		}
		else if(s.equals("August")){
			monthNum += "08";
		}
		else if(s.equals("September")){
			monthNum += "09";
		}
		else if(s.equals("October")){
			monthNum += "10";
		}
		else if(s.equals("November")){
			monthNum += "11";
		}
		else if(s.equals("December")){
			monthNum += "12";
		}
		return monthNum;
	}

	// Sort Months By Minimum Total Expenditure
	public static String selecExpSort(LinkedList a) {
		String b = "";
		Expenditure[] e = new Expenditure[Expenditure.totalEntries];

		for(int i = 0; i < a.size; i++) {
			e[i] = ((Expenditure) a.lookUp(i));
		}

		double min = -1;
		Expenditure temp;
		Expenditure tempmin = null;
		int pos = 0;
		for(int i = 0; i < a.size; i++) {
			for(int j = i; j < a.size; j++) {
				if(min == -1) {
					min = e[j].calculate();
					tempmin = e[j];
					pos = j;
				}
				else if(min > e[j].calculate()) {
					min = e[j].calculate();
					tempmin = e[j];
					pos = j;
```

```
                        }
                    }
                    temp = e[i];
                    e[pos] = temp;
                    e[i] = tempmin;

                    min = -1;
                }

                for(int i = 0; i < a.size; i++) {
                    if(i != a.size-1) b+=e[i].monthToString(e[i].getMonth(),
Expenditure.totalEntries) + ", ";
                    else b+=e[i].monthToString(e[i].getMonth(),
Expenditure.totalEntries);
                }
                return b;
        }

        // Sort Months By Maximum Total Expenditure
        public static String selecExpSortMax(LinkedList a) {
                String b = "";
                Expenditure[] e = new Expenditure[Expenditure.totalEntries];

                for(int i = 0; i < a.size; i++) {
                    e[i] = ((Expenditure) a.lookUp(i));
                }

                double max = -1;
                Expenditure temp;
                Expenditure tempmax = null;
                int pos = 0;
                for(int i = 0; i < a.size; i++) {
                    for(int j = i; j < a.size; j++) {
                        if(max == -1) {
                            max = e[j].calculate();
                            tempmax = e[j];
                            pos = j;
                        }
                        else if(max < e[j].calculate()) {
                            max = e[j].calculate();
                            tempmax = e[j];
                            pos = j;
                        }
                    }
                    temp = e[i];
                    e[pos] = temp;
                    e[i] = tempmax;

                    max = -1;
                }

                for(int i = 0; i < a.size; i++) {
                    if(i != a.size-1) b+=e[i].monthToString(e[i].getMonth(),
Expenditure.totalEntries) + ", ";
                    else b+=e[i].monthToString(e[i].getMonth(),
Expenditure.totalEntries);
                }
                return b;
        }
```

```
	// Sort Months By Minimum Total Income
	public static String selecIncSort(LinkedList a) {
		String b = "";
		Income[] e = new Income[Income.totalMonths];

		for(int i = 0; i < a.size; i++) {
			e[i] = ((Income) a.lookUp(i));
		}

		double min = -1;
		Income temp;
		Income tempmin = null;
		int pos = 0;
		for(int i = 0; i < a.size; i++) {
			for(int j = i; j < a.size; j++) {
				if(min == -1) {
					min = e[j].getIncome();
					tempmin = e[j];
					pos = j;
				}
				else if(min > e[j].getIncome()) {
					min = e[j].getIncome();
					tempmin = e[j];
					pos = j;
				}
			}
			temp = e[i];
			e[pos] = temp;
			e[i] = tempmin;

			min = -1;
		}

		for(int i = 0; i < a.size; i++) {
			if(i != a.size-1) b+=e[i].monthToString(e[i].getMonth(),
Income.totalMonths) + ", ";
			else b+=e[i].monthToString(e[i].getMonth(),
Income.totalMonths);
		}
		return b;
	}

	// Sort Months By Maximum Total Income
	public static String selecIncSortMax(LinkedList a) {
		String b = "";
		Income[] e = new Income[Income.totalMonths];

		for(int i = 0; i < a.size; i++) {
			e[i] = ((Income) a.lookUp(i));
		}

		double max = -1;
		Income temp;
		Income tempmax = null;
		int pos = 0;
		for(int i = 0; i < a.size; i++) {
			for(int j = i; j < a.size; j++) {
				if(max == -1) {
```

```
                                max = e[j].getIncome();
                                tempmax = e[j];
                                pos = j;
                        }
                        else if(max < e[j].getIncome()) {
                                max = e[j].getIncome();
                                tempmax = e[j];
                                pos = j;
                        }
                }
                temp = e[i];
                e[pos] = temp;
                e[i] = tempmax;

                max = -1;
        }

        for(int i = 0; i < a.size; i++) {
                if(i != a.size-1) b+=e[i].monthToString(e[i].getMonth(),
Income.totalMonths) + ", ";
                else b+=e[i].monthToString(e[i].getMonth(),
Income.totalMonths);
        }
        return b;
    }

    // Search Expenditure by Month
    public static int seqExpSearch(LinkedList b, String m) {
        for(int i = 0; i < b.size; i++) {
            if(((Expenditure) b.lookUp(i)).getMonth().equals(m)) return i;
        }
        return -1;
    }

    // Sort Income by Month
    public static int seqIncSearch(LinkedList b, String m) {
        for(int i = 0; i < b.size; i++) {
            if(((Income) b.lookUp(i)).getMonth().equals(m)) return i;
        }
        return -1;
    }
}

package ia;

import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class ImagePanel extends JPanel {
    private Image i;
    public ImagePanel(String i) {
        this(new ImageIcon(i).getImage());
    }

    public ImagePanel(Image i) {
        this.i = i;
```

```
		Dimension size = new Dimension(i.getWidth(null), i.getHeight(null));
		setPreferredSize(size);
		setMinimumSize(size);
		setMaximumSize(size);
		setSize(size);
		setLayout(null);
	}

	public void paintComponent(Graphics g) {
		g.drawImage(i, 0, 0, null);
	}
}

package ia;

import java.util.Scanner;
import java.io.*;

public class Income extends Money{
	static String currentMonth = "000000";
	private double taxRate;
	private double rawIncome;
	private double income;
	static String incomeInfo = "";
	static int totalMonths = 0;
	static File f = new File("TextFiles/" + LoginScreen.currentUser +
"info.txt");
	static int index = 0;

	public Income(String month, double taxRate, double rawIncome){
		super(month);
		this.taxRate = taxRate;
		this.rawIncome = rawIncome;
		calculate();
		if(currentMonth.equals("000000")) {
			currentMonth = month;
		}
		try {
			Scanner sc = new Scanner(f);
			while (sc.hasNextLine()) {
				String data = sc.nextLine();
				if(!sc.hasNextLine()){
					currentMonth = data.substring(0, data.indexOf('
'));
				}
				if(totalMonths == 0) {
					incomeInfo += data + "\n";
				}
			}
			sc.close();
		} catch (FileNotFoundException e) {
			  System.out.println("An error occurred.");
			  e.printStackTrace();
		}
		totalMonths++;
	}

	// Getter Methods
	public double getTaxRate(){
```

```
		return taxRate;
	}
	public double getRawIncome(){
		return rawIncome;
	}
	public double getIncome(){
		return income;
	}

	// Setter Methods
	public void setTaxRate(double taxRate){
		this.taxRate = taxRate;
	}
	public void setRawIncome(double rawIncome){
		this.rawIncome = rawIncome;
	}
	public void setIncome(double income){
		this.income = income;
	}

	// Find Income of Most Recent Entry
	public static double findCurrentIncome(){
		String lineMonth = "";
		double lineIncome = 0.0;
		double lineTax = 0.0;
		try {
			Scanner sc = new Scanner(f);
			while (sc.hasNextLine()) {
				String data = sc.nextLine();
				lineMonth = data.substring(0, data.indexOf(' '));
				lineIncome =
Double.parseDouble(data.substring(data.indexOf('-')+1,data.length()));
				lineTax =
Double.parseDouble(data.substring(data.indexOf(' ')+1, data.indexOf('-')));
				if(lineMonth.equals(currentMonth)){
					if(lineTax == 0.0) {
						return lineIncome;
					}
					else {
						return lineIncome - lineIncome *
(lineTax/100.0);
					}

				}
			}
			sc.close();
			return 0.0;
		} catch (FileNotFoundException e) {
			  System.out.println("An error occurred.");
			  e.printStackTrace();
		}
		return 0.0;
	}

	// Calculate Income
	public double calculate(){
		if(taxRate == 0.0) this.income = rawIncome;
		else this.income = rawIncome - rawIncome * (taxRate/100.0);
		return this.income;
```

```
	}

	// Update Strings
	public static void updateAll() {
		incomeInfo = incomeInfo.substring(0,incomeInfo.lastIndexOf(' '))+"
"+((Income) Main.i.lookUp(totalMonths -1)).getTaxRate()+"-"+((Income)
Main.i.lookUp(totalMonths -1)).getRawIncome() ;
	}
}

package ia;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class IncomeUpdater extends JFrame {

    private JButton jButton1;
    private JButton jButton2;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JTextField jTextField1;
    private JTextField jTextField2;
    private JTextField jTextField3;
       private JOptionPane jOptionPane1;
       private ImagePanel iuPanel = new ImagePanel(new
ImageIcon("./Images/iu.png").getImage());

    public IncomeUpdater() {
        initComponents();
    }

    private void initComponents() {

        jTextField1 = new JTextField();
        jTextField2 = new JTextField();
        jTextField3 = new JTextField();
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();
        jLabel3 = new JLabel();
        jButton1 = new JButton();
        jButton2 = new JButton();

        jLabel1.setFont(new Font("Arial", 1, 14));
        jLabel1.setText("Tax Rate:");

        jLabel2.setFont(new Font("Arial", 1, 14));
        jLabel2.setText("Monthly Income:");

        jLabel3.setFont(new Font("Arial", 1, 14));
        jLabel3.setText("Manually Change Income:");

             // Initializing the text boxes
             if(Income.totalMonths != 0){
                    for(int i = 0; i < Main.exp.size; i++) {
                           String mData = ((Income) Main.i.lookUp(i)).getMonth();
```

```
                        if(Income.currentMonth.equals(mData)) {
                                jTextField1.setText(((Income)
Main.i.lookUp(i)).getRawIncome()+"");
                                jTextField2.setText(((Income)
Main.i.lookUp(i)).getTaxRate()+"");
                                Income.index = i;
                                break;
                        }
                    }
             }

        jButton1.setText("Exit");
             jButton1.setBackground(new Color(102,178,255));
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

             jButton2.setBackground(new Color(102,178,255));
        jButton2.setText("Update");
             jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE, 202,
GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
107, Short.MAX_VALUE)
                        .addComponent(jButton2))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel2)
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jButton1))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1)
                            .addComponent(jTextField3, GroupLayout.PREFERRED_SIZE,
202, GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel3)
                            .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE,
202, GroupLayout.PREFERRED_SIZE))
                        .addGap(0, 0, Short.MAX_VALUE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
```

```
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(35, 35, 35)
                        .addComponent(jButton1)
                        .addGap(13, 13, 13)
                        .addComponent(jButton2))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(51, 51, 51)
                        .addComponent(jLabel2, GroupLayout.PREFERRED_SIZE, 17,
GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE, 32,
GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addComponent(jLabel1, GroupLayout.PREFERRED_SIZE, 12,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE, 32,
GroupLayout.PREFERRED_SIZE)
                .addGap(31, 31, 31)
                .addComponent(jLabel3)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField3, GroupLayout.PREFERRED_SIZE, 29,
GroupLayout.PREFERRED_SIZE)
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

             this.add(iuPanel);
             this.setIconImage(Main.icon.getImage());
             this.setTitle("SmartCash - Update Income Data");
        pack();
    }

    // Exit
    private void jButton1ActionPerformed(ActionEvent evt) {
        this.setVisible(false);
    }

    // Update
       private void jButton2ActionPerformed(ActionEvent evt) {
        double rawInc = 0.0, taxRate = 0.0, manual = 0.0;
             boolean success = true;
             if(jTextField3.getText().equals("")){
                    try{
                           rawInc = Double.parseDouble(jTextField1.getText());
                    }
                    catch(NumberFormatException e){
                           jTextField1.setText("Invalid value. Please try again.");
                           success = false;
                    }
                    try{
                           taxRate = Double.parseDouble(jTextField2.getText());
                    }
                    catch(NumberFormatException e){
                           jTextField2.setText("Invalid value. Please try again.");
                           success = false;
                    }
```

```
                    if(success){
                           ((Income)
Main.i.lookUp(Income.index)).setTaxRate(taxRate);
                           ((Income)
Main.i.lookUp(Income.index)).setRawIncome(rawInc);

                           if(Income.index == Income.totalMonths -1) {
                                  Income.incomeInfo =
Income.incomeInfo.substring(0,Income.incomeInfo.lastIndexOf(' ')) + " " + taxRate +
"-" + rawInc + "\n";
                                  rewrite();
                           }
                           else {
                                  rewriteFromScratch();
                           }
                           jOptionPane1 = new JOptionPane();
                           JFrame f = new JFrame();
                           JOptionPane.showMessageDialog(f,"You have successfully
updated this month's information.");
                           jTextField1.setText("");
                           jTextField2.setText("");

                           ((Income) Main.i.lookUp(Income.index)).calculate();
                           Menu.reload();
                    }
             }
             else{
                    try{
                           manual = Double.parseDouble(jTextField3.getText());
                           rawInc = manual;
                           ((Income)
Main.i.lookUp(Income.index)).setTaxRate(taxRate);
                           ((Income)
Main.i.lookUp(Income.index)).setRawIncome(manual);

                           if(Income.index == Income.totalMonths - 1) {
                                  Income.incomeInfo =
Income.incomeInfo.substring(0,Income.incomeInfo.lastIndexOf(' ')) + " " + taxRate +
"-" + rawInc + "\n";
                                  rewrite();
                           }
                           else {
                                  rewriteFromScratch();
                           }

                           jOptionPane1 = new JOptionPane();
                           JFrame f = new JFrame();
                           JOptionPane.showMessageDialog(f,"You have successfully
updated this month's information.");
                           jTextField1.setText("");
                           jTextField2.setText("");
                           ((Income)
Main.i.lookUp(Income.totalMonths-1)).calculate();
                           Menu.reload();
                    }
                    catch(NumberFormatException e){
                           jTextField3.setText("Invalid value. Please try again.");
                    }
             }
```

```
            this.setVisible(false);
    }

       // Rewrite By Updating Strings
       private void rewrite(){
              try {
                     FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "infoExp.txt");
                     BufferedWriter br = new BufferedWriter(fw);
                     br.write(Expenditure.expInfo);
                     br.close();
              } catch (FileNotFoundException e) {
                     System.out.println("An error occurred.");
                     e.printStackTrace();
              } catch (IOException e) {
                     System.out.println("An error occurred.");
                     e.printStackTrace();
              }
              try {
                     FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "info.txt");
                     BufferedWriter br = new BufferedWriter(fw);
                     br.write(Income.incomeInfo);
                     br.close();
              } catch (FileNotFoundException e) {
                     System.out.println("An error occurred.");
                     e.printStackTrace();
              } catch (IOException e) {
                     System.out.println("An error occurred.");
                     e.printStackTrace();
              }
       }

       // Rewrite by Recreating Strings
       private void rewriteFromScratch() {
              Expenditure.expInfo = "";
              Income.incomeInfo = "";
              for(int i = 0; i < Main.i.size; i++) {
                     Income.incomeInfo += ((Income) Main.i.lookUp(i)).getMonth() + "
" + ((Income) Main.i.lookUp(i)).getTaxRate() + "-" + ((Income)
Main.i.lookUp(i)).getRawIncome() + "\n";
              }
              for(int i = 0; i < Main.exp.size; i++) {
                     Expenditure.expInfo += ((Expenditure)
Main.exp.lookUp(i)).getMonth() + " " + "l" + ((Expenditure)
Main.exp.lookUp(i)).getLuxuryExp() + "f" + ((Expenditure)
Main.exp.lookUp(i)).getFoodExp() + "t" + ((Expenditure)
Main.exp.lookUp(i)).getTransportExp() + "u" + ((Expenditure)
Main.exp.lookUp(i)).getUtilityExp() + "o" + ((Expenditure)
Main.exp.lookUp(i)).getOtherExp() + "\n";
              }
              try {
                     FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "infoExp.txt");
                     BufferedWriter br = new BufferedWriter(fw);
                     br.write(Expenditure.expInfo);
                     br.close();
              } catch (FileNotFoundException e) {
```

```
                        System.out.println("An error occurred.");
                        e.printStackTrace();
                } catch (IOException e) {
                        System.out.println("An error occurred.");
                        e.printStackTrace();
                }
                try {
                        FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "info.txt");
                        BufferedWriter br = new BufferedWriter(fw);
                        br.write(Income.incomeInfo);
                        br.close();
                } catch (FileNotFoundException e) {
                        System.out.println("An error occurred.");
                        e.printStackTrace();
                } catch (IOException e) {
                        System.out.println("An error occurred.");
                        e.printStackTrace();
                }
        }
}


package ia;

public class Investment extends Money {
        private double totalSavings;
        private double simpleAmt = 0;
        private double compoundAmt = 0;
        private double simpleRate = 0;
        private double compoundRate = 0;
        private double investmentAmt = 0;

        public Investment (String month, double totalSavings) {
                super(month);
                this.totalSavings = totalSavings;
        }

        // Getter Methods
        public double getTotalSavings() {
                return this.totalSavings;
        }

        public double getInvestmentAmt() {
                return this.investmentAmt;
        }

        public double getCompoundInv() {
                return this.compoundAmt;
        }

        public double getSimpleInv() {
                return this.simpleAmt;
        }

        // Setter Methods
        public void setSimpleInv(double amount) {
                simpleAmt=amount;
                investmentAmt=compoundAmt+amount;
```

```
	}

	public void setCompoundInv(double amount) {
		compoundAmt=amount;
		investmentAmt=simpleAmt+amount;
	}

	public void setSimpleRate(double interestRate) {
		simpleRate = interestRate;
	}

	public void setCompoundRate(double interestRate) {
		compoundRate = interestRate;
	}

	// Calculations
	public double calculateSimpleReturn(int months) {
		return simpleAmt * (1 + simpleRate/100.0/12 * months);
	}

	public double calculateCompoundReturn(int months) {
		return compoundAmt * Math.pow(1 + (compoundRate/100.0/12), months);
	}
}
```

```
package ia;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class InvestmentMenu extends JFrame {
    static JButton jButton1;
    static JButton jButton5;
    static JButton jButton7;
    static JLabel jLabel10;
    static JLabel jLabel11;
    static JLabel jLabel12;
    static JLabel jLabel13;
    static JLabel jLabel3;
    static JLabel jLabel4;
    static JLabel jLabel7;
    static JLabel jLabel8;
    static JLabel jLabel9;
    static ImagePanel imPanel = new ImagePanel(new
ImageIcon("./Images/ic.png").getImage());
    static Investment inv = new Investment(Income.currentMonth, (((Income)
Main.i.lookUp(Income.index)).calculate() - ((Expenditure)
Main.exp.lookUp(Expenditure.index)).calculate()));

    public InvestmentMenu() {
        initComponents();
    }

    private void initComponents() {

        jLabel3 = new JLabel();
        jLabel4 = new JLabel();
```

```
        jButton1 = new JButton();
        jButton5 = new JButton();
        jButton7 = new JButton();
        jLabel7 = new JLabel();
        jLabel8 = new JLabel();
        jLabel9 = new JLabel();
        jLabel10 = new JLabel();
        jLabel11 = new JLabel();
        jLabel12 = new JLabel();
        jLabel13 = new JLabel();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jLabel3.setFont(Main.ft);
        String s = String.format("%.2f", inv.getTotalSavings());
        jLabel3.setText("Surplus Money This Month: $"+s);

        jLabel4.setFont(Main.ft);
        String f = String.format("%.2f", inv.getInvestmentAmt());
        jLabel4.setText("Total Money Invested: $"+f);

        jButton1.setText("Exit");
        jButton1.setBackground(new Color(102,178,255));
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton5.setText("Return to Main Menu");
        jButton5.setBackground(new Color(102,178,255));
        jButton5.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        jButton7.setText("Update");
        jButton7.setBackground(new Color(102,178,255));
        jButton7.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });

        jLabel7.setFont(Main.ft);
        jLabel7.setText("Expected Return:");

        jLabel8.setFont(Main.ft);
        String oneM = String.format("%.2f",
inv.calculateSimpleReturn(1)+inv.calculateCompoundReturn(1));
        jLabel8.setText("In 1 month: $" +oneM);

        jLabel9.setFont(Main.ft);
        String sixM = String.format("%.2f",
inv.calculateSimpleReturn(6)+inv.calculateCompoundReturn(6));
        jLabel9.setText("In 6 months: $"+sixM);

        jLabel10.setFont(Main.ft);
```

```
        String oneY = String.format("%.2f",
inv.calculateSimpleReturn(12)+inv.calculateCompoundReturn(12));
        jLabel10.setText("In 1 year: $"+oneY);

        jLabel11.setFont(Main.ft);
        String fiveY = String.format("%.2f",
inv.calculateSimpleReturn(60)+inv.calculateCompoundReturn(60));
        jLabel11.setText("In 5 years: $"+fiveY);

        jLabel12.setFont(Main.ft);
        String tenY= String.format("%.2f",
inv.calculateSimpleReturn(120)+inv.calculateCompoundReturn(120));
        jLabel12.setText("In 10 years: $"+tenY);

        jLabel13.setText("This application helps you envision potential investments
for you to make & compare them.");

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addContainerGap())
            .addGroup(layout.createSequentialGroup()
                .addGap(22, 22, 22)
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel4, GroupLayout.PREFERRED_SIZE,
412, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jButton7))
                            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel3,
GroupLayout.PREFERRED_SIZE, 597, GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel13,
GroupLayout.PREFERRED_SIZE, 700, GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel7,
GroupLayout.PREFERRED_SIZE, 498, GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING, false)
                                        .addComponent(jLabel11,
GroupLayout.Alignment.LEADING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                        .addComponent(jLabel10,
GroupLayout.Alignment.LEADING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                        .addComponent(jLabel9,
GroupLayout.Alignment.LEADING, GroupLayout.DEFAULT_SIZE, 473, Short.MAX_VALUE)
```

```
                                        .addComponent(jLabel8,
GroupLayout.Alignment.LEADING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                                    .addComponent(jLabel12,
GroupLayout.PREFERRED_SIZE, 463, GroupLayout.PREFERRED_SIZE))
                                .addGap(0, 0, Short.MAX_VALUE)))
                        .addContainerGap())
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jButton5, GroupLayout.PREFERRED_SIZE, 775,
GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 19, Short.MAX_VALUE))))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(41, 41, 41)
                .addComponent(jButton1)
                .addGap(77, 77, 77)
                .addComponent(jLabel3)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(jButton7))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel13)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton5, GroupLayout.PREFERRED_SIZE, 66,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel7)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel8)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel9)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel10)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel11)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel12)
                .addContainerGap(23, Short.MAX_VALUE))
        );
        this.setIconImage(Main.icon.getImage());
             this.setTitle("SmartCash - Investment Center");
             this.add(imPanel);
        pack();
    }

    // Exit
    private void jButton1ActionPerformed(ActionEvent evt) {
        this.setVisible(false);
    }

    // Open Investment Updater
    private void jButton7ActionPerformed(ActionEvent evt) {
       InvestmentUpdater iu = new InvestmentUpdater();
        iu.setVisible(true);
             iu.setLocationRelativeTo(null);
```

```
            iu.setResizable(false);
            iu.setDefaultCloseOperation(iu.DISPOSE_ON_CLOSE);
    }

    // Exit
    private void jButton5ActionPerformed(ActionEvent evt) {
       this.setVisible(false);
    }

    // Reload Menu
    public static void reload() {
        jLabel3.setFont(Main.ft);
        String s = String.format("%.2f", inv.getTotalSavings());
        jLabel3.setText("Surplus Money This Month: $"+s);

        jLabel4.setFont(Main.ft);
        String f = String.format("%.2f", inv.getInvestmentAmt());
        jLabel4.setText("Total Money Invested: $"+f);

        jLabel7.setFont(Main.ft);
        jLabel7.setText("Expected Return:");

        jLabel8.setFont(Main.ft);
        String oneM = String.format("%.2f",
inv.calculateSimpleReturn(1)+inv.calculateCompoundReturn(1));
        jLabel8.setText("In 1 month: $" +oneM);

        jLabel9.setFont(Main.ft);
        String sixM = String.format("%.2f",
inv.calculateSimpleReturn(6)+inv.calculateCompoundReturn(6));
        jLabel9.setText("In 6 months: $"+sixM);

        jLabel10.setFont(Main.ft);
        String oneY = String.format("%.2f",
inv.calculateSimpleReturn(12)+inv.calculateCompoundReturn(12));
        jLabel10.setText("In 1 year: $"+oneY);

        jLabel11.setFont(Main.ft);
        String fiveY = String.format("%.2f",
inv.calculateSimpleReturn(60)+inv.calculateCompoundReturn(60));
        jLabel11.setText("In 5 years: $"+fiveY);

        jLabel12.setFont(Main.ft);
        String tenY= String.format("%.2f",
inv.calculateSimpleReturn(120)+inv.calculateCompoundReturn(120));
        jLabel12.setText("In 10 years: $"+tenY);

        jLabel13.setText("This application helps you envision potential investments
for you to make & compare them.");
    }

}

package ia;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

```
public class InvestmentUpdater extends JFrame {
    private JButton jButton1;
    private JButton jButton2;
    private JButton jButton3;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JLabel jLabel4;
    private JTextField jTextField1;
    private JTextField jTextField2;
    private JTextField jTextField3;
    private JTextField jTextField4;
    private JOptionPane jOptionPane1;
    static ImagePanel iuPanel = new ImagePanel(new
ImageIcon("./Images/iu2.png").getImage());

    public InvestmentUpdater() {
        initComponents();
    }

    private void initComponents() {

        jLabel1 = new JLabel();
        jTextField1 = new JTextField();
        jLabel2 = new JLabel();
        jTextField2 = new JTextField();
        jLabel3 = new JLabel();
        jTextField3 = new JTextField();
        jTextField4 = new JTextField();
        jLabel4 = new JLabel();
        jButton1 = new JButton();
        jButton2 = new JButton();
        jButton3 = new JButton();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Total Simple Interest Investments");

        jLabel2.setText("Simple Interest Rate");

        jLabel3.setText("Total Compound Interest Investments");

        jLabel4.setText("Compound Interest Rate");

        jButton1.setText("Update");
        jButton1.setBackground(new Color(102,178,255));
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Update");
        jButton2.setBackground(new Color(102,178,255));
        jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
```

```
        });

        jButton3.setText("Exit");
        jButton3.setBackground(new Color(102,178,255));
        jButton3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING, false)
                                .addComponent(jTextField1)
                                .addComponent(jLabel1, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                            .addComponent(jLabel2))
                        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jButton1)
                            .addComponent(jButton2,
GroupLayout.Alignment.TRAILING)))
                    .addComponent(jLabel3)
                    .addComponent(jTextField3, GroupLayout.PREFERRED_SIZE, 161,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel4)
                    .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE, 161,
GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField4, GroupLayout.PREFERRED_SIZE, 161,
GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(94, 94, 94)
                        .addComponent(jButton3)))
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(30, 30, 30)
                .addComponent(jLabel1)
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                        .addGap(14, 14, 14)
                        .addComponent(jLabel2))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(23, 23, 23)
```

```
                        .addComponent(jButton1)))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                .addGap(26, 26, 26)
                .addComponent(jLabel3)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField3, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel4)
                    .addComponent(jButton2))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField4, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton3)
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        this.setIconImage(Main.icon.getImage());
              this.setTitle("SmartCash - Investment Center");
              this.add(iuPanel);
        pack();
    }

    // Update Simple Interest
    private void jButton1ActionPerformed(ActionEvent evt) {
       try {
              if(Double.parseDouble(jTextField1.getText()) +
InvestmentMenu.inv.getInvestmentAmt() > InvestmentMenu.inv.getTotalSavings() &&
Double.parseDouble(jTextField1.getText()) > InvestmentMenu.inv.getSimpleInv()) {
                    jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"You cannot invest more money
than you have saved.");
              }
              else {

InvestmentMenu.inv.setSimpleInv(Double.parseDouble(jTextField1.getText()));

InvestmentMenu.inv.setSimpleRate(Double.parseDouble(jTextField2.getText()));
                    InvestmentMenu.reload();
              }
       }
       catch(NumberFormatException e) {
              jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"Invalid data.");
       }
    }

    // Update Compound Interest
    private void jButton2ActionPerformed(ActionEvent evt) {
       try {
              if(Double.parseDouble(jTextField3.getText()) +
InvestmentMenu.inv.getInvestmentAmt() > InvestmentMenu.inv.getTotalSavings() &&
Double.parseDouble(jTextField3.getText()) > InvestmentMenu.inv.getCompoundInv()) {
```

```
                    jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"You cannot invest more money
than you have saved.");
             }
             else {

InvestmentMenu.inv.setCompoundInv(Double.parseDouble(jTextField3.getText()));

InvestmentMenu.inv.setCompoundRate(Double.parseDouble(jTextField4.getText()));
                    InvestmentMenu.reload();
             }
       }
       catch(NumberFormatException e) {
              jOptionPane1 = new JOptionPane();
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f,"Invalid data.");
       }
    }

    private void jButton3ActionPerformed(ActionEvent evt) {
        this.setVisible(false);
    }
}

package ia;

public class LinkedList implements List{
    Node header;
    int size;

    public LinkedList(){
        header = null;
        size = 0;
    }

    // Traverse Nodes
    private Node traverse(int i){
        Node n = header;
        if(i < 0) return null;

        for(int j = 0 ; j < i ; j++){
            if(n == null) return null;
            n = n.getNext();
        }
        return n;
    }

    // Insert Node
    public boolean insert(int i, Object item){
        Node prev;
        Node newNode = new Node(item);

        if(i == 0){
            if(!isEmpty())
               newNode.setNext(header);
            header = newNode;
        }
        else{
```

```
            prev = traverse(i - 1);
            if(prev == null)
               return false;
            newNode.setNext(prev.getNext());
            prev.setNext(newNode);
        }
        size++;
        return true;
    }

    // Check size
    public int size(){
        return size;
    }

    // Check if List is Empty
    public boolean isEmpty(){
        return header == null;
    }

    // Look Up Item by Index
    public Object lookUp(int i){
        Node n = traverse(i);
        if(n == null) return null;
        return n.getData();
    }

    // Delete a Node
    public boolean delete(int i){
        Node prev;
        if(isEmpty ()) return false;
        if(i == 0) header = header.getNext();
        else{
            prev = traverse (i - 1);
            if(prev.getNext () == null) return false;
            prev.setNext(prev.getNext().getNext());
        }
        size--;
        return true;
    }


    // Replace a Node
    public boolean replace(int i, Object item){
        Node n = traverse(i);
        if (n == null) return false;
        n.setData(item);
        return true;
    }

    public void displayReverse(){
        printReverse(header);
    }

   // Print Items in Reverse
    private void printReverse(Node head){
        if (head == null){}
        else{
            printReverse(head.getNext ());
```

```
            System.out.println((String) head.getData());
        }
    }
}


package ia;

public interface List{
    public boolean insert (int i, Object item);
    public int size ();
    public boolean isEmpty ();
    public Object lookUp (int i);
    public boolean delete (int i);
    public boolean replace (int i, Object item);
}

package ia;

import java.util.Scanner;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.imageio.ImageIO;
import javax.swing.*;

public class LoginScreen extends JFrame {
       private JButton jButton1;
    private JButton jButton2;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JPasswordField jPasswordField1;
    private JTextField jTextField1;
       private JOptionPane jOptionPane1;
       private ImagePanel lsPanel = new ImagePanel("./Images/ls2.png");
       static boolean firstLogin = true;
       static String loginInfo = "";
       private File logins = new File("./TextFiles/accounts.txt");
       static String currentUser = "";

       public LoginScreen() {
        initComponents();
              this.add(lsPanel);
              this.setIconImage(Main.icon.getImage());
              this.setTitle("SmartCash - Login");
              try {
                     Scanner sc = new Scanner(logins);
                     while (sc.hasNextLine()) {
                            loginInfo += sc.nextLine() + "\n";
                     }
                     sc.close();
              } catch (FileNotFoundException e) {
                       System.out.println("An error occurred.");
                       e.printStackTrace();
              }
    }

    private void initComponents() {
        jButton1 = new JButton();
```

```
        jButton2 = new JButton();
        jTextField1 = new JTextField();
        jPasswordField1 = new JPasswordField();
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jButton1.setText("Login");
             jButton1.setBackground(new Color(102,178,255));
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Register");
             jButton2.setBackground(new Color(102,178,255));
        jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jLabel1.setText("Username");

        jLabel2.setText("Password");

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(202, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING, false)
                        .addComponent(jPasswordField1)
                        .addComponent(jTextField1)
                        .addComponent(jButton2, GroupLayout.DEFAULT_SIZE, 162,
Short.MAX_VALUE)
                        .addComponent(jButton1, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addGap(203, 203, 203))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(159, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE, 34,
GroupLayout.PREFERRED_SIZE)
                .addGap(3, 3, 3)
                .addComponent(jLabel2)
```

```
                .addGap(1, 1, 1)
                .addComponent(jPasswordField1, GroupLayout.PREFERRED_SIZE, 34,
GroupLayout.PREFERRED_SIZE)
                .addGap(26, 26, 26)
                .addComponent(jButton1, GroupLayout.PREFERRED_SIZE, 31,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton2, GroupLayout.PREFERRED_SIZE, 31,
GroupLayout.PREFERRED_SIZE)
                .addGap(121, 121, 121))
        );

        pack();
    }


    // Create User-Specific Files
       private void createFile() {
               try {
                      File newFile = new File("./TextFiles/" +
LoginScreen.currentUser + "info.txt");
                      if (newFile.createNewFile()) {
                             System.out.println("The file was created.");
                      }
                      else {
                             System.out.println("The file already exists.");
                      }

                      File newFile2 = new File("./TextFiles/" +
LoginScreen.currentUser + "infoExp.txt");
                      if (newFile2.createNewFile()) {
                             System.out.println("The file was created.");
                      }
                      else {
                             System.out.println("The file already exists.");
                      }
               }
               catch(IOException e) {
                     System.out.println("File could not be created.");
               }
       }

       // Login Button
    private void jButton1ActionPerformed(ActionEvent evt) {
              String username = jTextField1.getText();
              String password = jPasswordField1.getText();
              boolean success = false;
              String existingUser = "";
              String existingPswd = "";
              try {
                     Scanner sc = new Scanner(logins);
                     while (sc.hasNextLine()) {
                             String data = sc.nextLine();
                             existingUser = data.substring(0, data.indexOf(' '));
                             existingPswd = data.substring(data.indexOf('
')+1,data.length());
                             if(username.equals(existingUser) &&
password.equals(existingPswd)){
                                    System.out.println("Successful Login");
```

```
                                success = true;
                                if(firstLogin){
                                       firstLogin = false;
                                       currentUser = username;
                                       createFile();
                                       Menu.menu();
                                }
                                else{
                                       if(currentUser != username){
                                              currentUser = username;
                                              // Switching Users
                                              try {

System.out.println(LoginScreen.currentUser);
                                                     File newFile = new
File("./TextFiles/" + LoginScreen.currentUser + "info.txt");
                                                     if (newFile.createNewFile()) {
                                                            System.out.println("The
file was created.");
                                                     }
                                                     else {
                                                            System.out.println("The
file already exists.");
                                                     }

                                                     File newFile2 = new
File("./TextFiles/" + LoginScreen.currentUser + "infoExp.txt");
                                                     if (newFile2.createNewFile())
{
                                                            System.out.println("The
file was created.");
                                                     }
                                                     else {
                                                            System.out.println("The
file already exists.");
                                                     }
                                              } catch (IOException e) {
                                                     System.out.println("An error
occurred.");
                                                     e.printStackTrace();
                                              }

                                              // Clearing the Object LinkedLists
                                              for(int i = 0; i <
Income.totalMonths; i++) {
                                                     Main.i.delete(0);
                                                     Main.exp.delete(0);
                                              }

                                              // Clearing the Static Variables
                                              Expenditure.currentMonth = "000000";
                                              Income.currentMonth = "000000";
                                              Expenditure.totalEntries = 0;
                                              Income.totalMonths = 0;
                                              Expenditure.expInfo = "";
                                              Income.incomeInfo = "";
```

```
                                                // Changing the target files
                                                Income.f = new File("./TextFiles/" +
LoginScreen.currentUser + "info.txt");
                                                Expenditure.f = new
File("./TextFiles/" + LoginScreen.currentUser + "infoExp.txt");

                                                // Recreating the Object ArrayLists
                                                Main.initLists();

                                                Menu.reload();
                                        }
                                        Menu.f.setVisible(true);
                                }
                                this.setVisible(false);
                                break;
                        }
                }
                if(!success){
                        jOptionPane1 = new JOptionPane();
                        JFrame f = new JFrame();
                        JOptionPane.showMessageDialog(f,"Invalid username or
password.");
                }
                sc.close();
            } catch (FileNotFoundException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
            }
    }

       // Open Register Screen
    private void jButton2ActionPerformed(ActionEvent evt) {
            RegisterScreen r = new RegisterScreen();
            r.setVisible(true);
            r.setLocationRelativeTo(null);
            r.setResizable(false);
    }
}

package ia;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Menu{
       static JOptionPane jOptionPane1;
       static JButton jButton1;
    static JButton jButton2;
    static JButton jButton3;
    static JButton jButton4;
    static JButton jButton5;
       static JButton jButton6;
       static JButton jButton7;
    static JLabel jLabel1;
    static JLabel jLabel2;
    static JLabel jLabel3;
    static JLabel jLabel4;
```

```
    static JLabel jLabel5;
    static JLabel jLabel6;
    static JScrollPane jScrollPane1;
    static JTextArea jTextArea1;
       static JFrame f = new JFrame();
       static ImagePanel mPanel = new ImagePanel(new
ImageIcon("./Images/menu.png").getImage());
       static String savingsAdvice = "";
       static String luxuryAdvice = "";
       static String utilityAdvice = "";
       static String foodAdvice = "";
       static String transportAdvice = "";
       static String otherAdvice = "";

       public static void menu(){
              CardLayout cl = new CardLayout();
              JPanel mainPanel = new JPanel();
              JPanel menuPanel = new JPanel();
              mainPanel.setLayout(cl);

              jLabel1 = new JLabel();
         jLabel2 = new JLabel();
         jLabel3 = new JLabel();
         jLabel4 = new JLabel();
         jLabel5 = new JLabel();
         jButton1 = new JButton();
         jButton2 = new JButton();
         jButton3 = new JButton();
         jButton4 = new JButton();
         jButton5 = new JButton();
         jScrollPane1 = new JScrollPane();
         jTextArea1 = new JTextArea();
         jLabel6 = new JLabel();
         jButton6 = new JButton();
         jButton7 = new JButton();

              Income.f = new File("./TextFiles/" + LoginScreen.currentUser +
"info.txt");
              Expenditure.f = new File("./TextFiles/" + LoginScreen.currentUser +
"infoExp.txt");
              Main.initLists(); // Creating the array lists of Income & Expenditure
Objects
              Income.index = Income.totalMonths - 1; // Initializing the index
variable
              Expenditure.index = Expenditure.totalEntries - 1; // Initializing the
index variable

              // GUI
         jLabel1.setFont(Main.ft);
         jLabel1.setText("Welcome back, "+ LoginScreen.currentUser + ". Month: " +
Expenditure.monthToString(Expenditure.currentMonth, Expenditure.totalEntries));

         jLabel3.setFont(Main.ft);
              if(Income.findCurrentIncome() < 0){
                     jLabel3.setText("Monthly Income: $0 [Enter data]");
              }
              else {
                     String s = Income.findCurrentIncome() + "";
                     jLabel3.setText("Monthly Income: $"+s);
```

```
            }
            jLabel4.setFont(Main.ft);
            if(Expenditure.calcExp() <= 0){
                  jLabel4.setText("Monthly Expenses: $0 [Enter data]");
            }
            else {
                  String s = String.format("%.2f", Expenditure.calcExp());
                  jLabel4.setText("Monthly Expenses: $"+s);
            }

        jLabel5.setFont(Main.ft);
            if(Income.findCurrentIncome() > 0 && Expenditure.calcExp() > 0){
                  String s = String.format("%.2f", Income.findCurrentIncome() -
Expenditure.calcExp());
                  jLabel5.setText("Surplus Money: $" + s);
            }
            else if(Income.findCurrentIncome() > 0){
                  String s = String.format("%.2f", Income.findCurrentIncome());
                  jLabel5.setText("Surplus Money: $" + s);
            }
            else{
                  jLabel5.setText("Surplus Money: $0 [Enter Data]");
            }

        jButton1.setText("Exit");
             jButton1.setBackground(new Color(0,76,153));
             jButton1.setForeground(Color.WHITE);
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("New Month");
             jButton2.setBackground(new Color(102,178,255));
        jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton3.setText("Past Months");
             jButton3.setBackground(new Color(102,178,255));
        jButton3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jButton4.setText("Change Past Data");
             jButton4.setBackground(new Color(102,178,255));
        jButton4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton5.setText("Savings & Investment Center");
             jButton5.setBackground(new Color(102,178,255));
```

```
        jButton5.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });
             jButton6.setText("Update");
             jButton6.setBackground(new Color(102,178,255));
        jButton6.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
        });

        jButton7.setText("Update");
             jButton7.setBackground(new Color(102,178,255));
             jButton7.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });
        jTextArea1.setColumns(20);
        jTextArea1.setRows(4);
        jScrollPane1.setViewportView(jTextArea1);
        jTextArea1.setEditable(false);

        jLabel6.setFont(Main.ft);
        jLabel6.setText("Personalized Advice");

        if(Income.totalMonths != 0) {
              if((((Income) Main.i.lookUp(Income.index)).getIncome() -
((Expenditure) Main.exp.lookUp(Expenditure.index)).calculate()) / ((Income)
Main.i.lookUp(Income.index)).getIncome() > 0.15) {
                    savingsAdvice = "You are saving a good amount of money.";
              }
              else {
                    savingsAdvice = "You should be saving more money. It is
recommended to save anywhere from 15-30% of your income. Around half of that should
be invested.";
              }

              if(((Expenditure) Main.exp.lookUp(Expenditure.index)).getLuxuryExp()
/ ((Income) Main.i.lookUp(Income.index)).getIncome() > 0.1) {
                    luxuryAdvice = "You are spending too much money on luxury
goods. You should be spending under 10% of your income on luxury goods.";
              }
              else {
                    luxuryAdvice = "You are spending a healthy portion of your
income on luxury goods!";
              }

              if(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getUtilityExp() / ((Income)
Main.i.lookUp(Income.index)).getIncome() > 0.20) {
                    utilityAdvice = "Your utility bills are too expensive. They
should be taking up under 20% of your monthly income.";
              }
              else {
                    utilityAdvice = "You are spending a healthy portion of your
income on utility bills!";
```

```
            }

            if(((Expenditure) Main.exp.lookUp(Expenditure.index)).getFoodExp() >
0) {
                  foodAdvice = "Each person you are supporting should cost around
$300 per month or less in terms of food expenditure.";
            }
            else {
                  foodAdvice = "Please enter food expenditure data.";
            }

            if(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getTransportExp()  > 0) {
                  utilityAdvice = "You should be spending at most $180 on
transport per month per person.";
            }
            else {
                  utilityAdvice = "Please enter transport expenditure data!";
            }
        }

        jTextArea1.setText(savingsAdvice + "\n" + luxuryAdvice + "\n" +
utilityAdvice + "\n" + foodAdvice + "\n" + transportAdvice + "\n" + otherAdvice +
"\n");


        GroupLayout layout = new GroupLayout(menuPanel);
        menuPanel.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(22, 22, 22)
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(jLabel2, GroupLayout.PREFERRED_SIZE, 500,
GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jScrollPane1, GroupLayout.PREFERRED_SIZE,
775, GroupLayout.PREFERRED_SIZE)
                            .addComponent(jButton5, GroupLayout.PREFERRED_SIZE,
775, GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jButton2, GroupLayout.PREFERRED_SIZE,
256, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jButton3, GroupLayout.PREFERRED_SIZE,
275, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jButton4, GroupLayout.PREFERRED_SIZE,
232, GroupLayout.PREFERRED_SIZE)))
                        .addGap(0, 21, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
```

```
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel4, GroupLayout.PREFERRED_SIZE,
500, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jButton7))
                            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel1,
GroupLayout.PREFERRED_SIZE, 500, GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel5,
GroupLayout.PREFERRED_SIZE, 500, GroupLayout.PREFERRED_SIZE))
                                .addGap(0, 0, Short.MAX_VALUE))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel3, GroupLayout.PREFERRED_SIZE,
500, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jButton6)))
                        .addContainerGap())))
            .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addComponent(jButton1)
                        .addContainerGap())
                    .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addComponent(jLabel6, GroupLayout.PREFERRED_SIZE, 500,
GroupLayout.PREFERRED_SIZE)
                        .addGap(296, 296, 296))))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(41, 41, 41)
                .addComponent(jButton1)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
                .addGap(45, 45, 45)
                .addComponent(jLabel1, GroupLayout.PREFERRED_SIZE, 30,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel3)
                    .addComponent(jButton6))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(jButton7))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel5)
```

```
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton2)
                    .addComponent(jButton3)
                    .addComponent(jButton4))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton5, GroupLayout.PREFERRED_SIZE, 66,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel6)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane1, GroupLayout.PREFERRED_SIZE, 170,
GroupLayout.PREFERRED_SIZE)
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

              menuPanel.add(mPanel);
              mainPanel.add(menuPanel, "1");
              cl.show(mainPanel, "1");

              // JFrame
              f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
              f.pack();
              f.add(mainPanel);
              f.setTitle("SmartCash - Financial Tracking Application");
              f.setIconImage(Main.icon.getImage());
              f.setSize(835,640);
              f.setResizable(false);
              f.setLocationRelativeTo(null); // Center frame on screen
              f.setVisible(true);

       }

       // Exit, Return to Login Screen
       public static void jButton1ActionPerformed(ActionEvent evt) {
              f.setVisible(false);
              LoginScreen ls = new LoginScreen();
              ls.setVisible(true);
         ls.setLocationRelativeTo(null);
    }

       // Open MonthCreator
    public static void jButton2ActionPerformed(ActionEvent evt) {
              MonthCreator mc = new MonthCreator();
              mc.setVisible(true);
              mc.setLocationRelativeTo(null);
              mc.setResizable(false);
              mc.setDefaultCloseOperation(mc.DISPOSE_ON_CLOSE);
    }

    // Open History Viewer
    public static void jButton3ActionPerformed(ActionEvent evt) {
              HistoryViewer hv = new HistoryViewer();
              hv.setVisible(true);
              hv.setLocationRelativeTo(null);
              hv.setResizable(false);
              hv.setDefaultCloseOperation(hv.DISPOSE_ON_CLOSE);
    }
```

```
    // Open ChangePastData Menu
    public static void jButton4ActionPerformed(ActionEvent evt) {
       ChangePastData cpd = new ChangePastData();
       cpd.setVisible(true);
       cpd.setLocationRelativeTo(null);
       cpd.setResizable(false);
       cpd.setDefaultCloseOperation(cpd.DISPOSE_ON_CLOSE);
    }

    // Open InvestmentMenu
    public static void jButton5ActionPerformed(ActionEvent evt) {
       InvestmentMenu im = new InvestmentMenu();
              im.setVisible(true);
              im.setLocationRelativeTo(null);
              im.setResizable(false);
              im.setDefaultCloseOperation(im.DISPOSE_ON_CLOSE);
    }

    // Open Income Updater
       public static void jButton6ActionPerformed(ActionEvent evt) {
              if(Income.totalMonths == 0){
                     jOptionPane1 = new JOptionPane();
                     JFrame f = new JFrame();
                     JOptionPane.showMessageDialog(f,"Create a new month first.");
              }
              else{
                     IncomeUpdater iu = new IncomeUpdater();
                     iu.setVisible(true);
                     iu.setLocationRelativeTo(null);
                     iu.setResizable(false);
                     iu.setDefaultCloseOperation(iu.DISPOSE_ON_CLOSE);
              }
    }

    // Open Expenditure Updater
       public static void jButton7ActionPerformed(ActionEvent evt) {
              if(Expenditure.totalEntries == 0){
                     jOptionPane1 = new JOptionPane();
                     JFrame f = new JFrame();
                     JOptionPane.showMessageDialog(f,"Create a new month first.");
              }
              else{
                     ExpUpdater eu = new ExpUpdater();
                     eu.setVisible(true);
                     eu.setLocationRelativeTo(null);
                     eu.setResizable(false);
                     eu.setDefaultCloseOperation(eu.DISPOSE_ON_CLOSE);
                     eu.setSize(440,410);
              }
    }

       // Reload Screen
       public static void reload(){
              jLabel1.setFont(Main.ft);
        jLabel1.setText("Welcome back, "+ LoginScreen.currentUser + ". Month: " +
Expenditure.monthToString(Expenditure.currentMonth, Expenditure.totalEntries));

              jLabel3.setFont(Main.ft);
```

```
            if(Income.findCurrentIncome() < 0){
                  jLabel3.setText("Monthly Income: $0 [Enter data]");
            }
            else {
                  String s = Income.findCurrentIncome() + "";
                  jLabel3.setText("Monthly Income: $"+s);
            }
            jLabel4.setFont(Main.ft);
            if(Expenditure.calcExp() <= 0){
                  jLabel4.setText("Monthly Expenses: $0 [Enter data]");
            }
            else {
                  String s = String.format("%.2f", Expenditure.calcExp());
                  jLabel4.setText("Monthly Expenses: $"+s);
            }

        jLabel5.setFont(Main.ft);
            if(Income.findCurrentIncome() > 0 && Expenditure.calcExp() > 0){
                  String s = String.format("%.2f", Income.findCurrentIncome() -
Expenditure.calcExp());
                  jLabel5.setText("Surplus Money: $" + s);
            }
            else if(Income.findCurrentIncome() > 0){
                  String s = String.format("%.2f", Income.findCurrentIncome());
                  jLabel5.setText("Surplus Money: $" + s);
            }
            else{
                  jLabel5.setText("Surplus Money: $0 [Enter Data]");
            }

        if(Income.totalMonths != 0) {
            Income.index = Income.totalMonths - 1;
            Expenditure.index = Expenditure.totalEntries - 1;

             if((((Income) Main.i.lookUp(Income.index)).getIncome() -
((Expenditure) Main.exp.lookUp(Expenditure.index)).calculate()) / ((Income)
Main.i.lookUp(Income.index)).getIncome() > 0.15) {
                  savingsAdvice = "You are saving a good amount of money.";
             }
             else {
                  savingsAdvice = "You should be saving more money. It is
recommended to save anywhere from 15-30% of your income. Around half of that should
be invested.";
             }

             if(((Expenditure) Main.exp.lookUp(Expenditure.index)).getLuxuryExp()
/ ((Income) Main.i.lookUp(Income.index)).getIncome() > 0.1) {
                  luxuryAdvice = "You are spending too much money on luxury
goods. You should be spending under 10% of your income on luxury goods.";
             }
             else {
                  luxuryAdvice = "You are spending a healthy portion of your
income on luxury goods!";
             }

             if(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getUtilityExp() / ((Income)
Main.i.lookUp(Income.index)).getIncome() > 0.20) {
```

```
                    utilityAdvice = "Your utility bills are too expensive. They
should be taking up under 20% of your monthly income.";
               }
               else {
                    utilityAdvice = "You are spending a healthy portion of your
income on utility bills!";
               }

               if(((Expenditure) Main.exp.lookUp(Expenditure.index)).getFoodExp() >
0) {
                    foodAdvice = "Each person you are supporting should cost around
$300 per month or less in terms of food expenditure.";
               }
               else {
                    foodAdvice = "Please enter food expenditure data.";
               }

               if(((Expenditure)
Main.exp.lookUp(Expenditure.index)).getTransportExp()  > 0) {
                    utilityAdvice = "You should be spending at most $180 on
transport per month per person.";
               }
               else {
                    utilityAdvice = "Please enter transport expenditure data!";
               }
               jTextArea1.setText(savingsAdvice + "\n" + luxuryAdvice + "\n" +
utilityAdvice + "\n" + foodAdvice + "\n" + transportAdvice + "\n" + otherAdvice +
"\n");
         }
       }
}


package ia;

public class Money {
       private String month;

       public Money(String month) {
              this.month = month;
       }

       public String getMonth() {
              return this.month;
       }

       public double calculate() {
              return Double.parseDouble(month);
       }

       public static String monthToString(String month, int total){
              if(total > 0){
                     String s = "";

                     String m = month.substring(0, 2);
                     String y = month.substring(2, month.length());

                     if(m.equals("01")){
                            s += "January ";
```

```
                }
                else if(m.equals("02")){
                       s += "February ";
                }
                else if(m.equals("03")){
                       s += "March ";
                }
                else if(m.equals("04")){
                       s += "April ";
                }
                else if(m.equals("05")){
                       s += "May ";
                }
                else if(m.equals("06")){
                       s += "June ";
                }
                else if(m.equals("07")){
                       s += "July ";
                }
                else if(m.equals("08")){
                       s += "August ";
                }
                else if(m.equals("09")){
                       s += "September ";
                }
                else if(m.equals("10")){
                       s += "October ";
                }
                else if(m.equals("11")){
                       s += "November ";
                }
                else if(m.equals("12")){
                       s += "December ";
                }

                       return s + y + "";
             }
             return "Add a month";
       }
}
```

```
package ia;

import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MonthCreator extends JFrame {
    private JButton jButton1;
    private JButton jButton2;
    private JComboBox<String> jComboBox1;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JTextField jTextField1;
       private JOptionPane jOptionPane1;
       private ImagePanel mcPanel = new ImagePanel(new
ImageIcon("./Images/nmt.png").getImage());
```

```
    public MonthCreator() {
        initComponents();
    }

    private void initComponents() {

        jComboBox1 = new JComboBox<>();
        jLabel1 = new JLabel();
        jTextField1 = new JTextField();
        jLabel2 = new JLabel();
        jButton1 = new JButton();
        jButton2 = new JButton();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jComboBox1.setModel(new DefaultComboBoxModel<>(new String[] { "January",
"February", "March", "April", "May", "June", "July", "August", "September",
"October", "December" }));

        jLabel1.setText("Select Month");
        jLabel2.setText("Enter Year");

        jButton1.setText("Create New Month");
             jButton1.setBackground(new Color(102,178,255));
             jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Exit");
             jButton2.setBackground(new Color(102,178,255));
             jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addComponent(jComboBox1, GroupLayout.PREFERRED_SIZE,
249, GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel1))
                        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
                            .addComponent(jLabel2, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jTextField1)))
                    .addGroup(layout.createSequentialGroup()
```

```
                        .addGap(144, 144, 144)
                        .addComponent(jButton2)))
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addGap(106, 106, 106))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(46, 46, 46)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jLabel2))
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                    .addComponent(jComboBox1, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jButton1)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton2)
                .addContainerGap(16, Short.MAX_VALUE))
        );

             this.setIconImage(Main.icon.getImage());
             this.setTitle("SmartCash - Add New Entry");
             this.add(mcPanel);
        pack();
    }

    // Add Month
    private void jButton1ActionPerformed(ActionEvent evt) {
             String month = jComboBox1.getSelectedItem() + "";
        month = monthToNum(month);
             try{
                    int year = Integer.parseInt(jTextField1.getText());
                    int mostRecentYr =
Integer.parseInt(Expenditure.currentMonth.substring(2));
                    int mostRecentMo =
Integer.parseInt(Expenditure.currentMonth.substring(0,2));

                    if(jTextField1.getText().equals("")){
                           jOptionPane1 = new JOptionPane();
                           JFrame f = new JFrame();
                           JOptionPane.showMessageDialog(f,"Please enter the
year.");
                    }
                    else{
                           int index = -1;
                           boolean valid = true;
                           for(int i = 0; i < Main.exp.size; i++) {
```

```
                                String mData = ((Expenditure)
Main.exp.lookUp(i)).getMonth().substring(0,2);
                                int yData = Integer.parseInt(((Expenditure)
Main.exp.lookUp(i)).getMonth().substring(2));
                                if(year == yData) {
                                      if(Integer.parseInt(month) ==
Integer.parseInt(mData)) {
                                            jOptionPane1 = new JOptionPane();
                                            JFrame f = new JFrame();

JOptionPane.showMessageDialog(f,"Data for that month already exists.");
                                            valid = false;
                                            break;
                                      }
                                }
                          }

                          if(valid) {
                                if(Income.totalMonths <= 1) index =
Income.totalMonths;
                                else if(Income.totalMonths > 1){
                                      for(int i = 0; i < Main.exp.size; i++) {
                                            String mData = ((Expenditure)
Main.exp.lookUp(i)).getMonth().substring(0,2);
                                            int yData =
Integer.parseInt(((Expenditure) Main.exp.lookUp(i)).getMonth().substring(2));
                                            if(year < yData) {
                                                  System.out.println("if");
                                                  index = i;
                                                  break;
                                            }
                                            else if(year == yData && i <
Main.exp.size-1) {
                                                  if(Integer.parseInt(month) >
Integer.parseInt(mData) && Integer.parseInt(month) <
Integer.parseInt(((Expenditure) Main.exp.lookUp(i+1)).getMonth().substring(0,2))) {

System.out.println("else if");
                                                        index = i+1;
                                                        break;
                                                  }
                                            }
                                            else if(year == yData && i ==
Main.exp.size-1) {
                                                  System.out.println("else if
2");
                                                  if(Integer.parseInt(month) >
Integer.parseInt(mData) && year > Integer.parseInt(((Expenditure)
Main.exp.lookUp(i-1)).getMonth().substring(2))) {
                                                        index = i+1;
                                                        break;
                                                  }
                                                  else
if(Integer.parseInt(month) > Integer.parseInt(mData)) {
                                                        index = i+1;
                                                        break;
                                                  }
                                            }
                                            else if(year > yData) {
```

```
                                                System.out.println("else");
                                                index = i+1;
                                        }
                                }
                        }
                        if(index == -1) index = Income.totalMonths;
                        System.out.println(index);
                        month += year + "";
                        if(Expenditure.totalEntries > 0 &&
Income.totalMonths > 0 && index == Income.totalMonths){
                                Expenditure.expInfo += month + " " + "l" +
0.0 + "f" + 0.0 + "t" + 0.0 + "u" + 0.0 + "o" + 0.0 + "\n";
                                Income.incomeInfo += month + " " +
((Income) Main.i.lookUp(Income.totalMonths-1)).getTaxRate() + "-" + ((Income)
Main.i.lookUp(Income.totalMonths-1)).getRawIncome() + "\n";

                                Main.exp.insert(Expenditure.totalEntries,
new Expenditure(month, 0.0, 0.0, 0.0, 0.0, 0.0));
                                Main.i.insert(Income.totalMonths, new
Income(month, ((Income) Main.i.lookUp(Income.totalMonths-1)).getTaxRate(),
((Income) Main.i.lookUp(Income.totalMonths-1)).getRawIncome()));

                                Income.currentMonth = month;
                                Expenditure.currentMonth = month;
                                rewrite();
                        }
                        else if(index == Income.totalMonths){
                                Expenditure.expInfo += month + " " + "l" +
0.0 + "f" + 0.0 + "t" + 0.0 + "u" + 0.0 + "o" + 0.0 + "\n";
                                Income.incomeInfo += month + " " + 0.0 +
"-" + 0.0 + "\n";

                                Main.exp.insert(Expenditure.totalEntries,
new Expenditure(month, 0.0, 0.0, 0.0, 0.0, 0.0));
                                Main.i.insert(Income.totalMonths, new
Income(month, 0.0, 0.0));

                                Income.currentMonth = month;
                                Expenditure.currentMonth = month;
                                rewrite();
                        }
                        else {
                                Main.exp.insert(index, new
Expenditure(month, 0.0, 0.0, 0.0, 0.0, 0.0));
                                Main.i.insert(index, new Income(month, 0.0,
0.0));

                                rewriteFromScratch();
                        }

                        System.out.println(index);

                        jOptionPane1 = new JOptionPane();
                        JFrame f = new JFrame();
                        JOptionPane.showMessageDialog(f,"You have
successfully created a new month! Please enter expenditure data or update income
information.");
                        Menu.reload();
                        this.setVisible(false);
```

```
                        }
                }
        }
        catch(NumberFormatException e){
                jTextField1.setText("Invalid value. Please try again.");
        }
}

 // Exit
   private void jButton2ActionPerformed(ActionEvent evt) {
    this.setVisible(false);
}

   private String monthToNum(String s){
        String monthNum = "";

        if(s.equals("January")){
                monthNum += "01";
        }
        else if(s.equals("February")){
                monthNum += "02";
        }
        else if(s.equals("March")){
                monthNum += "03";
        }
        else if(s.equals("April")){
                monthNum += "04";
        }
        else if(s.equals("May")){
                monthNum += "05";
        }
        else if(s.equals("June")){
                monthNum += "06";
        }
        else if(s.equals("July")){
                monthNum += "07";
        }
        else if(s.equals("August")){
                monthNum += "08";
        }
        else if(s.equals("September")){
                monthNum += "09";
        }
        else if(s.equals("October")){
                monthNum += "10";
        }
        else if(s.equals("November")){
                monthNum += "11";
        }
        else if(s.equals("December")){
                monthNum += "12";
        }
        return monthNum;
   }


   // Rewrite Methods
   private void rewrite(){
        try {
```

```
                    FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "infoExp.txt");
                    BufferedWriter br = new BufferedWriter(fw);
                    br.write(Expenditure.expInfo);
                    br.close();
             } catch (FileNotFoundException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             } catch (IOException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             }
             try {
                    FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "info.txt");
                    BufferedWriter br = new BufferedWriter(fw);
                    br.write(Income.incomeInfo);
                    br.close();
             } catch (FileNotFoundException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             } catch (IOException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             }
      }

      private void rewriteFromScratch() {
             Expenditure.expInfo = "";
             Income.incomeInfo = "";
             for(int i = 0; i < Main.i.size; i++) {
                    Income.incomeInfo += ((Income) Main.i.lookUp(i)).getMonth() + "
" + ((Income) Main.i.lookUp(i)).getTaxRate() + "-" + ((Income)
Main.i.lookUp(i)).getRawIncome() + "\n";
             }
             for(int i = 0; i < Main.exp.size; i++) {
                    Expenditure.expInfo += ((Expenditure)
Main.exp.lookUp(i)).getMonth() + " " + "l" + ((Expenditure)
Main.exp.lookUp(i)).getLuxuryExp() + "f" + ((Expenditure)
Main.exp.lookUp(i)).getFoodExp() + "t" + ((Expenditure)
Main.exp.lookUp(i)).getTransportExp() + "u" + ((Expenditure)
Main.exp.lookUp(i)).getUtilityExp() + "o" + ((Expenditure)
Main.exp.lookUp(i)).getOtherExp() + "\n";
             }
             try {
                    FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "infoExp.txt");
                    BufferedWriter br = new BufferedWriter(fw);
                    br.write(Expenditure.expInfo);
                    br.close();
             } catch (FileNotFoundException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             } catch (IOException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             }
             try {
```

```
                    FileWriter fw = new FileWriter("./TextFiles/" +
LoginScreen.currentUser + "info.txt");
                    BufferedWriter br = new BufferedWriter(fw);
                    br.write(Income.incomeInfo);
                    br.close();
             } catch (FileNotFoundException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             } catch (IOException e) {
                    System.out.println("An error occurred.");
                    e.printStackTrace();
             }
       }
}
```

```
package ia;

public class Node{
    private Object data;
    private Node next;

    public Node(){
        this (null, null);
    }

    public Node(Object data, Node next){
        this.data = data;
        this.next = next;
    }

    public Node(Object data){
        this.data = data;
        this.next = null;
    }

    public void setData(Object data){
        this.data = data;
    }

    public void setNext(Node next){
        this.next = next;
    }

    public Object getData(){
        return data;
    }

    public Node getNext(){
        return next;
    }
}
```

```
package ia;

import java.util.Scanner;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.imageio.ImageIO;
```

```
import javax.swing.*;

public class RegisterScreen extends JFrame {
    private JButton jButton1;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JPasswordField jPasswordField1;
    private JPasswordField jPasswordField2;
    private JTextField jTextField1;
       private JOptionPane jOptionPane1;
       private ImagePanel rsPanel;
       private File logins = new File("./TextFiles/accounts.txt");

    public RegisterScreen() {
        initComponents();
        rsPanel = new ImagePanel("./Images/rs.png");
             this.add(rsPanel);
             this.setIconImage(Main.icon.getImage());
             this.setDefaultCloseOperation(this.DISPOSE_ON_CLOSE);
    }

    private void initComponents() {

        jTextField1 = new JTextField();
        jButton1 = new JButton();
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();
        jLabel3 = new JLabel();
        jPasswordField1 = new JPasswordField();
        jPasswordField2 = new JPasswordField();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jButton1.setText("Register");
             jButton1.setBackground(new Color(102,178,255));
             jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel1.setText("Create Username");

        jLabel2.setText("Create Password");

        jLabel3.setText("Re-Enter Password");


        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(41, 41, 41)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
```

```
                            .addComponent(jTextField1)
                            .addComponent(jLabel1)
                            .addComponent(jLabel2)
                            .addComponent(jLabel3)
                            .addComponent(jPasswordField1)
                            .addComponent(jPasswordField2,
GroupLayout.DEFAULT_SIZE, 308, Short.MAX_VALUE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(131, 131, 131)
                        .addComponent(jButton1, GroupLayout.PREFERRED_SIZE, 118,
GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(51, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(35, 35, 35)
                .addComponent(jLabel1)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE, 29,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPasswordField1, GroupLayout.PREFERRED_SIZE, 32,
GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel3)
                .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPasswordField2, GroupLayout.PREFERRED_SIZE, 32,
GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton1, GroupLayout.PREFERRED_SIZE, 33,
GroupLayout.PREFERRED_SIZE)
                .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
             this.setIconImage(Main.icon.getImage());
             this.setTitle("SmartCash - Register");
        pack();
    }

    // Register Button
       private void jButton1ActionPerformed(ActionEvent evt) {
         String username = jTextField1.getText();
              String password = jPasswordField1.getText();
              String password2 = jPasswordField2.getText();

              if(!password.equals(password2)){
                     jOptionPane1 = new JOptionPane();
                     JFrame f = new JFrame();
                     JOptionPane.showMessageDialog(f,"Your passwords don't match.");
              }
              else{
                     boolean isValid = true;

                     if(!(username.indexOf(' ') < 0)){
                            jOptionPane1 = new JOptionPane();
                            JFrame f = new JFrame();
```

```
                        JOptionPane.showMessageDialog(f,"You cannot have spaces
in your username.");
                    }
                    else if(password.equals("") || username.equals("")){
                        jOptionPane1 = new JOptionPane();
                        JFrame f = new JFrame();
                        JOptionPane.showMessageDialog(f,"Your username/password
cannot be empty.");
                    }
                    else{
                        String existingUser = "";
                        try {
                            Scanner sc = new Scanner(logins);
                            while (sc.hasNextLine()) {
                                String data = sc.nextLine();
                                existingUser = data.substring(0,
data.indexOf(' '));
                                if(username.equals(existingUser)){
                                    jOptionPane1 = new JOptionPane();
                                    JFrame f = new JFrame();
                                    System.out.println("Username
Taken");

JOptionPane.showMessageDialog(f,"That username has been taken already.");
                                    isValid = false;
                                    break;
                                }
                            }
                            if(isValid){
                                try {
                                    LoginScreen.loginInfo += username +
" " + password + "\n";
                                    FileWriter fw = new
FileWriter("./TextFiles/accounts.txt");
                                    BufferedWriter br = new
BufferedWriter(fw);
                                    br.write(LoginScreen.loginInfo);
                                    br.close();

                                    jOptionPane1 = new JOptionPane();
                                    JFrame f = new JFrame();
                                    JOptionPane.showMessageDialog(f,"You
have successfully created an account. You may sign in now.");
                                    this.setVisible(false);
                                } catch (FileNotFoundException e) {
                                    System.out.println("An error
occurred.");
                                    e.printStackTrace();
                                } catch (IOException e) {
                                    System.out.println("An error
occurred.");
                                    e.printStackTrace();
                                }
                            }
                            sc.close();
                        } catch (FileNotFoundException e) {
                            System.out.println("An error occurred.");
                            e.printStackTrace();
                        }
```

```
                    }
                }
            }
        }
```