```
package loanassistant;

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

 public class LoanAssistant extends JFrame
 {
       JLabel balanceLabel = new JLabel();
       JTextField balanceTextField = new JTextField();
       JLabel interestLabel = new JLabel();
       JTextField interestTextField = new JTextField();
       JLabel monthsLabel = new JLabel();
       JTextField monthsTextField = new JTextField();
       JLabel paymentLabel = new JLabel();
       JTextField paymentTextField = new JTextField(); JButton
       computeButton = new JButton();
       JButton newLoanButton = new JButton();
       JButton monthsButton = new JButton();
       JButton paymentButton = new JButton();
       JLabel analysisLabel = new JLabel();
       JTextArea analysisTextArea = new JTextArea();
       JButton exitButton = new JButton();
       Font myFont = new Font("Arial", Font.PLAIN, 16); Color
       lightYellow = new Color(255, 255, 128);
       boolean computePayment;
       public static void main(String args[])
       {
       // create frame
       new LoanAssistant().~~show()~~;
       }
       public LoanAssistant()
       {
       // frame constructor
       setTitle("Loan Assistant");
       setResizable(false);
       addWindowListener(new WindowAdapter()
       {
       public void windowClosing(WindowEvent evt)
       {
       exitForm(evt);
       }
       });
       getContentPane().setLayout(new GridBagLayout());
       GridBagConstraints gridConstraints;
       balanceLabel.setText("Loan Balance");
       balanceLabel.setFont(myFont);
       gridConstraints = new GridBagConstraints();
       gridConstraints.gridx = 0;
       gridConstraints.gridy = 0;
       gridConstraints.anchor = GridBagConstraints.WEST;
       gridConstraints.insets = new Insets(10, 10, 0, 0);
       getContentPane().add(balanceLabel, gridConstraints);
       balanceTextField.setPreferredSize(new Dimension(100, 25));
balanceTextField.setHorizontalAlignment(SwingConstants.RIGHT);
balanceTextField.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 1;
gridConstraints.gridy = 0;
gridConstraints.insets = new Insets(10, 10, 0, 10);
getContentPane().add(balanceTextField, gridConstraints);
balanceTextField.addActionListener(new ActionListener ()
{
public void actionPerformed(ActionEvent e)
{
balanceTextFieldActionPerformed(e);
} });
interestLabel.setText("Interest Rate");
interestLabel.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 0;
gridConstraints.gridy = 1;
gridConstraints.anchor = GridBagConstraints.WEST;
gridConstraints.insets = new Insets(10, 10, 0, 0);
getContentPane().add(interestLabel, gridConstraints);
interestTextField.setPreferredSize(new Dimension(100, 25));
interestTextField.setHorizontalAlignment(SwingConstants.RIGHT);
interestTextField.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 1;
gridConstraints.gridy = 1;
gridConstraints.insets = new Insets(10, 10, 0, 10);
getContentPane().add(interestTextField, gridConstraints);
interestTextField.addActionListener(new ActionListener ()
{
public void actionPerformed(ActionEvent e)
{
interestTextFieldActionPerformed(e);
} });
monthsLabel.setText("Number of Payments");
monthsLabel.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 0;
gridConstraints.gridy = 2;
gridConstraints.anchor = GridBagConstraints.WEST;
gridConstraints.insets = new Insets(10, 10, 0, 0);
getContentPane().add(monthsLabel, gridConstraints);
monthsTextField.setPreferredSize(new Dimension(100, 25));
monthsTextField.setHorizontalAlignment(SwingConstants.RIGHT);
monthsTextField.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 1;
gridConstraints.gridy = 2;
gridConstraints.insets = new Insets(10, 10, 0, 10);
getContentPane().add(monthsTextField, gridConstraints);
monthsTextField.addActionListener(new ActionListener ()
{
public void actionPerformed(ActionEvent e)
{
```

```
monthsTextFieldActionPerformed(e);
}
});
paymentLabel.setText("Monthly Payment");
paymentLabel.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 0;
gridConstraints.gridy = 3;
gridConstraints.anchor = GridBagConstraints.WEST;
gridConstraints.insets = new Insets(10, 10, 0, 0);
getContentPane().add(paymentLabel, gridConstraints);
paymentTextField.setPreferredSize(new Dimension(100, 25));
paymentTextField.setHorizontalAlignment(SwingConstants.RIGHT);
paymentTextField.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 1;
gridConstraints.gridy = 3;
gridConstraints.insets = new Insets(10, 10, 0, 10);
getContentPane().add(paymentTextField, gridConstraints);
paymentTextField.addActionListener(new ActionListener ()
{
public void actionPerformed(ActionEvent e)
{
paymentTextFieldActionPerformed(e);
}
});
computeButton.setText("Compute Monthly Payment");
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 0;
gridConstraints.gridy = 4;
gridConstraints.gridwidth = 2;
gridConstraints.insets = new Insets(10, 0, 0, 0);
getContentPane().add(computeButton, gridConstraints);
computeButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
computeButtonActionPerformed(e);
}
});
newLoanButton.setText("New Loan Analysis");
newLoanButton.setEnabled(false);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 0;
gridConstraints.gridy = 5;
gridConstraints.gridwidth = 2;
gridConstraints.insets = new Insets(10, 0, 10, 0);
getContentPane().add(newLoanButton, gridConstraints);
newLoanButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
newLoanButtonActionPerformed(e);
}
});
```

```java
monthsButton.setText("X");
monthsButton.setFocusable(false);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 2;
gridConstraints.gridy = 2;
gridConstraints.insets = new Insets(10, 0, 0, 0);
getContentPane().add(monthsButton, gridConstraints);
monthsButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
monthsButtonActionPerformed(e);
}
});
paymentButton.setText("X");
paymentButton.setFocusable(false);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 2;
gridConstraints.gridy = 3;
gridConstraints.insets = new Insets(10, 0, 0, 0);
getContentPane().add(paymentButton, gridConstraints);
paymentButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
paymentButtonActionPerformed(e);
}
});
analysisLabel.setText("Loan Analysis:");
analysisLabel.setFont(myFont);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 3;
gridConstraints.gridy = 0;
gridConstraints.anchor = GridBagConstraints.WEST;
gridConstraints.insets = new Insets(0, 10, 0, 0);
getContentPane().add(analysisLabel, gridConstraints);
analysisTextArea.setPreferredSize(new Dimension(250,
150));
analysisTextArea.setFocusable(false);
analysisTextArea.setBorder(BorderFactory.createLineBorder(
Color.BLACK));
analysisTextArea.setFont(new Font("Courier New",
Font.PLAIN, 14));
analysisTextArea.setEditable(false);
analysisTextArea.setBackground(Color.WHITE);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 3;
gridConstraints.gridy = 1;
gridConstraints.gridheight = 4;
gridConstraints.insets = new Insets(0, 10, 0, 10);
getContentPane().add(analysisTextArea, gridConstraints);
exitButton.setText("Exit");
exitButton.setFocusable(false);
gridConstraints = new GridBagConstraints();
gridConstraints.gridx = 3;
```

```
        gridConstraints.gridy = 5;
        getContentPane().add(exitButton, gridConstraints);
        exitButton.addActionListener(new ActionListener()
        {
        public void actionPerformed(ActionEvent e)
        {
        exitButtonActionPerformed(e);
        }
        });
        pack();
        Dimension screenSize =
        Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((int) (0.5 * (screenSize.width - getWidth())), (int)
        (0.5 * (screenSize.height - getHeight())), getWidth(),
        getHeight());
        paymentButton.doClick();
        }
        private void exitForm(WindowEvent evt)
        {
        System.exit(0);
        }
        private void computeButtonActionPerformed(ActionEvent e)
        {
        double balance, interest, payment;
        int months;
        double monthlyInterest, multiplier;
        double loanBalance, finalPayment;
        if (validateDecimalNumber(balanceTextField))
        {
        balance =
        Double.valueOf(balanceTextField.getText()).doubleValue();
        }
        else
        {
        JOptionPane.showConfirmDialog(null, "Invalid or empty Loan Balance
entry.\nPlease correct.", "Balance Input Error",
        JOptionPane.DEFAULT_OPTION,
        JOptionPane.INFORMATION_MESSAGE);
        return;
        }
        if (validateDecimalNumber(interestTextField))
        {
        interest =
        Double.valueOf(interestTextField.getText()).doubleValue();
        }
        else
        {
        JOptionPane.showConfirmDialog(null, "Interest Rate entry.\nPlease correct.",
"Interest Input Error",
        JOptionPane.DEFAULT_OPTION,
        JOptionPane.INFORMATION_MESSAGE);
        return;
        }
        monthlyInterest = interest / 1200; if (computePayment)
        {
```

```
        // Compute loan payment
        if (validateDecimalNumber(monthsTextField))
        {
        months =
        Integer.valueOf(monthsTextField.getText()).intValue();
        }
        else
        {
        JOptionPane.showConfirmDialog(null, "Invalid or empty Number of Payments
entry.\nPlease correct.", "Number of Payments Input Error",
                     JOptionPane.DEFAULT_OPTION,
        JOptionPane.INFORMATION_MESSAGE);
        return;
        }
        if (interest == 0)
        {
        payment = balance / months;
        }
        else
        {
        multiplier = Math.pow(1 + monthlyInterest, months);
        payment = balance * monthlyInterest * multiplier /
        (multiplier - 1);
        }
        paymentTextField.setText(new
        DecimalFormat("0.00").format(payment));
        }
        else
        {
        // Compute number of payments
        if (validateDecimalNumber(paymentTextField))
        {
        payment =
        Double.valueOf(paymentTextField.getText()).doubleValue();
        if (payment <= (balance * monthlyInterest + 1.0))
        {
        if (JOptionPane.showConfirmDialog(null, "Minimum paymentmust be $" + new
DecimalFormat("0.00").format((int)(balance * monthlyInterest + 1.0)) + "\n" + "Do you
want to use the minimum payment?", "Input Error",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE) ==
        JOptionPane.YES_OPTION)
        {
        paymentTextField.setText(new
        DecimalFormat("0.00").format((int)(balance *
        monthlyInterest + 1.0)));
        payment =
        Double.valueOf(paymentTextField.getText()).doubleValue();
        }
        else
        {
        paymentTextField.requestFocus();
        return;
        }
        }
```

```
      }
      else
      {
      JOptionPane.showConfirmDialog(null, "Invalid or empty Monthly Payment
entry.\nPlease correct.","Payment Input Error", JOptionPane.DEFAULT_OPTION,
      JOptionPane.INFORMATION_MESSAGE);
      return;
      }
      if (interest == 0)
      { months = (int)(balance / payment);
      }
      else
      {
      months = (int)((Math.log(payment) - Math.log(payment -
      balance * monthlyInterest)) / Math.log(1 + monthlyInterest));
      }
      monthsTextField.setText(String.valueOf(months));
      }
      // reset payment prior to analysis to fix at two decimals
      payment =
      Double.valueOf(paymentTextField.getText()).doubleValue();
      // show analysis
      analysisTextArea.setText("Loan Balance: $" + new
      DecimalFormat("0.00").format(balance));
      analysisTextArea.append("\n" + "Interest Rate: " + new
      DecimalFormat("0.00").format(interest) + "%");
      // process all but last payment
      loanBalance = balance;
      for (int paymentNumber = 1; paymentNumber <= months -
      1; paymentNumber++)
      {
      loanBalance += loanBalance * monthlyInterest - payment;
      }
      // find final payment
      finalPayment = loanBalance;
      if (finalPayment > payment)
      {
      // apply one more payment
      loanBalance += loanBalance * monthlyInterest - payment;
      finalPayment = loanBalance;
      months++;
      monthsTextField.setText(String.valueOf(months));
      }
      analysisTextArea.append("\n\n" + String.valueOf(months - 1)
      + " Payments of $" + new
      DecimalFormat("0.00").format(payment));
      analysisTextArea.append("\n" + "Final Payment of: $" + new
      DecimalFormat("0.00").format(finalPayment));
      analysisTextArea.append("\n" + "Total Payments: $" + new
      DecimalFormat("0.00").format((months - 1) * payment +
      finalPayment));
      analysisTextArea.append("\n" + "Interest Paid $" + new
      DecimalFormat("0.00").format((months - 1) * payment +
      finalPayment - balance));
      computeButton.setEnabled(false);
```

```java
newLoanButton.setEnabled(true);
newLoanButton.requestFocus();
}
private void newLoanButtonActionPerformed(ActionEvent e)
{
// clear computed value and analysis
if (computePayment)
{
paymentTextField.setText("");
}
else
{ monthsTextField.setText("");
}
analysisTextArea.setText("");
computeButton.setEnabled(true);
newLoanButton.setEnabled(false);
balanceTextField.requestFocus();
}
private void monthsButtonActionPerformed(ActionEvent e)
{
// will compute months
computePayment = false;
paymentButton.setVisible(true);
monthsButton.setVisible(false);
monthsTextField.setText("");
monthsTextField.setEditable(false);
monthsTextField.setBackground(lightYellow);
monthsTextField.setFocusable(false);
paymentTextField.setEditable(true);
paymentTextField.setBackground(Color.WHITE);
paymentTextField.setFocusable(true);
computeButton.setText("Compute Number of Payments");
balanceTextField.requestFocus();
}
private void paymentButtonActionPerformed(ActionEvent e)
{
// will compute payment
computePayment = true;
paymentButton.setVisible(false);
monthsButton.setVisible(true);
monthsTextField.setEditable(true);
monthsTextField.setBackground(Color.WHITE);
monthsTextField.setFocusable(true);
paymentTextField.setText("");
paymentTextField.setEditable(false);
paymentTextField.setBackground(lightYellow);
paymentTextField.setFocusable(false);
computeButton.setText("Compute Monthly Payment");
balanceTextField.requestFocus();
}
private void exitButtonActionPerformed(ActionEvent e)
{
System.exit(0);
}
private void balanceTextFieldActionPerformed(ActionEvent
e)
{
balanceTextField.transferFocus();
}
private void interestTextFieldActionPerformed(ActionEvent
e)
{
interestTextField.transferFocus();
}
private void monthsTextFieldActionPerformed(ActionEvent
e)
{
monthsTextField.transferFocus();
}
private void paymentTextFieldActionPerformed(ActionEvent
e)
{
paymentTextField.transferFocus();
}
private boolean validateDecimalNumber(JTextField tf)
{
// checks to see if text field contains
// valid decimal number with only digits and a single decimal
//point
String s = tf.getText().trim();
boolean hasDecimal = false;
boolean valid = true;
if (s.length() == 0)
{
valid = false;
}
else
{
for (int i = 0; i < s.length(); i++)
{
char c = s.charAt(i);
if (c >= '0' && c <= '9')
{
continue;
}
else if (c == '.' && !hasDecimal)
{
hasDecimal = true;
}
else
{
// invalid character found
valid = false;
}
}
}
tf.setText(s);
if (!valid)
{
tf.requestFocus();
```

```
        }
        return (valid);
        }
}
```