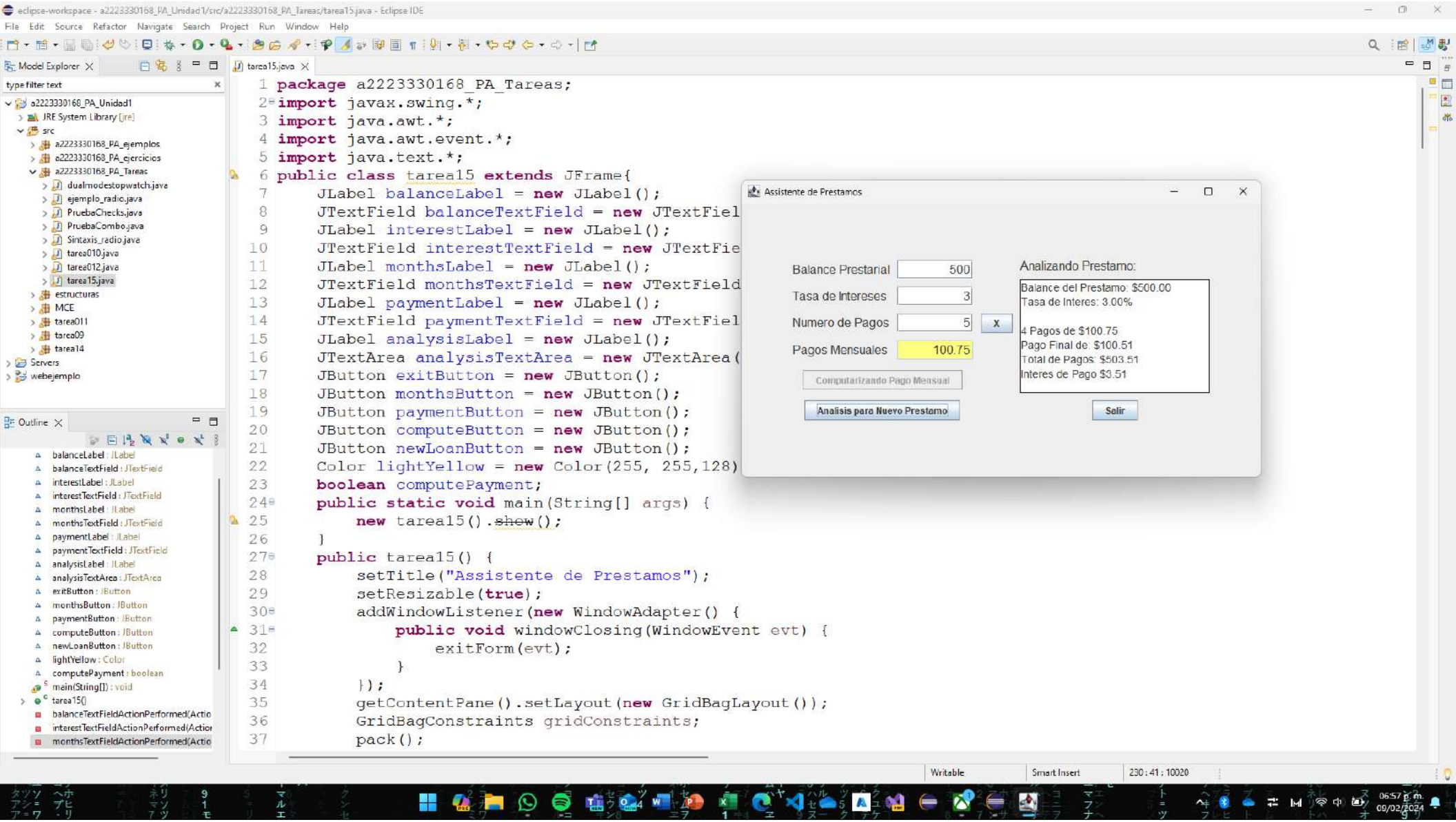
eclipse-workspace - a2223330168_PA_Unidad1/src/a2223330168_PA_Tareas/tarea15.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Model Explorer
type filter text
a2223330168_PA_Unidad1
JRE System Library [jre]
src
a2223330168_PA_ejemplos
a2223330168_PA_ejercicios
a2223330168_PA_Tareas
dualmodestopwatch.java
ejemplo_radio.java
PruebaChecks.java
PruebaCombo.java
Sintaxis_radio.java
tarea010.java
tarea012.java
tarea15.java
estructuras
MCE
tarea011
tarea09
tarea14
Servers
webejemplo
Outline
balanceLabel : JLabel
balanceTextField : JTextField
interestLabel : JLabel
interestTextField : JTextField
monthsLabel : JLabel
monthsTextField : JTextField
paymentLabel : JLabel
paymentTextField : JTextField
analysisLabel : JLabel
analysisTextArea : JTextArea
exitButton : JButton
monthsButton : JButton
paymentButton : JButton
computeButton : JButton
newLoanButton : JButton
lightYellow : Color
computePayment : boolean
main(String[]) : void
tarea15()
balanceTextFieldActionPerformed(Actio
interestTextFieldActionPerformed(Actior
monthsTextFieldActionPerformed(Actio
tarea15.java
1 package a2223330168_PA_Tareas;
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.*;
5 import java.text.*;
6 public class tarea15 extends JFrame{
7 JLabel balanceLabel = new JLabel();
8 JTextField balanceTextField = new JTextFiel
9 JLabel interestLabel = new JLabel();
10 JTextField interestTextField = new JTextFie
11 JLabel monthsLabel = new JLabel();
12 JTextField monthsTextField = new JTextField
13 JLabel paymentLabel = new JLabel();
14 JTextField paymentTextField = new JTextFiel
15 JLabel analysisLabel = new JLabel();
16 JTextArea analysisTextArea = new JTextArea(
17 JButton exitButton = new JButton();
18 JButton monthsButton = new JButton();
19 JButton paymentButton = new JButton();
20 JButton computeButton = new JButton();
21 JButton newLoanButton = new JButton();
22 Color lightYellow = new Color(255, 255,128)
23 boolean computePayment;
24 public static void main(String[] args) {
25 new tarea15().show();
26 }
27 public tarea15() {
28 setTitle("Assistente de Prestamos");
29 setResizable(true);
30 addWindowListener(new WindowAdapter() {
31 public void windowClosing(WindowEvent evt) {
32 exitForm(evt);
33 }
34 });
35 getContentPane().setLayout(new GridBagLayout());
36 GridBagConstraints gridConstraints;
37 pack();
Assistente de Prestamos
Balance Prestarial
500
Tasa de Intereses
3
Numero de Pagos
5
X
Pagos Mensuales
100.75
Computarizando Pago Mensual
Analisis para Nuevo Prestamo
Analizando Prestamo:
Balance del Prestamo: $500.00
Tasa de Interes: 3.00%
4 Pagos de $100.75
Pago Final de: $100.51
Total de Pagos: $503.51
Interes de Pago $3.51
Salir
Writable
Smart Insert
230 : 41 : 10020
06:57 p. m.
09/02/2024

```java
package a2223330168_PA_Tareas;
import javax.swing.*;
public class tarea15 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();
    JLabel analysisLabel = new JLabel();
    JTextArea analysisTextArea = new JTextArea();
    JButton exitButton = new JButton();
    JButton monthsButton = new JButton();
    JButton paymentButton = new JButton();
    JButton computeButton = new JButton();
    JButton newLoanButton = new JButton();
    Color lightYellow = new Color(255, 255,128);
    boolean computePayment;
    public static void main(String[] args) {
        new tarea15().show();
    }
    public tarea15() {
        setTitle("Assistente de Prestamos");
```

```
29          setResizable(true);
30          addWindowListener(new WindowAdapter() {
31              public void windowClosing(WindowEvent evt) {
32                  exitForm(evt);
33              }
34          });
35          getContentPane().setLayout(new GridBagLayout());
36          GridBagConstraints gridConstraints;
37          pack();
38          Dimension screenSize =
  Toolkit.getDefaultToolkit().getScreenSize();
39          setBounds((int) (0.5 * (screenSize.width - getWidth())), (int)
  (0.5 * (screenSize.height - getHeight())), getWidth(), getHeight());
40
41          Font myFont = new Font("Arial", Font.PLAIN, 16);
42
43          balanceLabel.setText("Balance Prestarial");
44          balanceLabel.setFont(myFont);
45          gridConstraints = new GridBagConstraints();
46          gridConstraints.gridx = 0;gridConstraints.gridy = 0;
47          gridConstraints.anchor = GridBagConstraints.WEST;
48          gridConstraints.insets = new Insets(10, 10, 0, 0);
49          getContentPane().add(balanceLabel, gridConstraints);
50          balanceTextField.setPreferredSize(new Dimension(100, 25));
51          balanceTextField.setHorizontalAlignment(SwingConstants.RIGHT);
```

```
52          balanceTextField.setFont(myFont);
53          balanceTextField.addActionListener(new ActionListener() {
54              public void actionPerformed(ActionEvent e) {
55                  balanceTextFieldActionPerformed(e);
56              }
57          });
58          gridConstraints = new GridBagConstraints();
59          gridConstraints.gridx = 1;gridConstraints.gridy = 0;
60          gridConstraints.insets = new Insets(10, 10, 0, 10);
61          getContentPane().add(balanceTextField, gridConstraints);
62          interestLabel.setText("Tasa de Intereses");
63          interestLabel.setFont(myFont);
64          gridConstraints = new GridBagConstraints();
65          gridConstraints.gridx = 0;gridConstraints.gridy = 1;
66          gridConstraints.anchor = GridBagConstraints.WEST;
67          gridConstraints.insets = new Insets(10, 10, 0, 0);
68          getContentPane().add(interestLabel, gridConstraints);
69          interestTextField.setPreferredSize(new Dimension(100, 25));
70          interestTextField.setHorizontalAlignment(SwingConstants.RIGHT);
71          interestTextField.setFont(myFont);
72          interestTextField.addActionListener(new ActionListener() {
73              public void actionPerformed(ActionEvent e) {
74                  interestTextFieldActionPerformed(e);
75              }
76          });
```

```
 77         gridConstraints = new GridBagConstraints();
 78         gridConstraints.gridx = 1;
 79         gridConstraints.gridy = 1;
 80         gridConstraints.insets = new Insets(10, 10, 0, 10);
 81         getContentPane().add(interestTextField, gridConstraints);
 82         monthsLabel.setText("Numero de Pagos");
 83         monthsLabel.setFont(myFont);
 84         gridConstraints = new GridBagConstraints();
 85         gridConstraints.gridx = 0;
 86         gridConstraints.gridy = 2;
 87         gridConstraints.anchor = GridBagConstraints.WEST;
 88         gridConstraints.insets = new Insets(10, 10, 0, 0);
 89         getContentPane().add(monthsLabel, gridConstraints);
 90         monthsTextField.setPreferredSize(new Dimension(100, 25));
 91         monthsTextField.setHorizontalAlignment(SwingConstants.RIGHT);
 92         monthsTextField.setFont(myFont);
 93         monthsTextField.addActionListener(new ActionListener() {
 94             public void actionPerformed(ActionEvent e) {
 95                 monthsTextFieldActionPerformed(e);
 96             }
 97         });
 98         gridConstraints = new GridBagConstraints();
 99         gridConstraints.gridx = 1;
100         gridConstraints.gridy = 2;
101         gridConstraints.insets = new Insets(10, 10, 0, 10);
```

```
102         getContentPane().add(monthsTextField, gridConstraints);
103         paymentLabel.setText("Pagos Mensuales");
104         paymentLabel.setFont(myFont);
105         gridConstraints = new GridBagConstraints();
106         gridConstraints.gridx = 0;
107         gridConstraints.gridy = 3;
108         gridConstraints.anchor = GridBagConstraints.WEST;
109         gridConstraints.insets = new Insets(10, 10, 0, 0);
110         getContentPane().add(paymentLabel, gridConstraints);
111         paymentTextField.setPreferredSize(new Dimension(100, 25));
112         paymentTextField.setHorizontalAlignment(SwingConstants.RIGHT);
113         paymentTextField.setFont(myFont);
114         paymentTextField.addActionListener(new ActionListener() {
115             public void actionPerformed(ActionEvent e) {
116                 paymentTextFieldActionPerformed(e);
117             }
118         });
119         gridConstraints = new GridBagConstraints();
120         gridConstraints.gridx = 1;
121         gridConstraints.gridy = 3;
122         gridConstraints.insets = new Insets(10, 10, 0, 10);
123         getContentPane().add(paymentTextField, gridConstraints);
124
125
126
```

```
127         computeButton.setText("Pago Computarizado Mensualmente");
128         gridConstraints = new GridBagConstraints();
129         gridConstraints.gridx = 0;
130         gridConstraints.gridy = 4;
131         gridConstraints.gridwidth = 2;
132         gridConstraints.insets = new Insets(10, 0, 0, 0);
133         getContentPane().add(computeButton, gridConstraints);
134         computeButton.addActionListener(new ActionListener()
135         {
136             public void actionPerformed(ActionEvent e)
137             {
138                 computeButtonActionPerformed(e);
139             }
140         });
141         newLoanButton.setText("Analisis para Nuevo Prestamo");
142         newLoanButton.setEnabled(false);
143         gridConstraints = new GridBagConstraints();
144         gridConstraints.gridx = 0;
145         gridConstraints.gridy = 5;
146         gridConstraints.gridwidth = 2;
147         gridConstraints.insets = new Insets(10, 0, 10, 0);
148         getContentPane().add(newLoanButton, gridConstraints);
149         newLoanButton.addActionListener(new ActionListener() {
150             public void actionPerformed(ActionEvent e) {
151                 newLoanButtonActionPerformed(e);
```

```
152             }
153         });
154 
155 
156 
157         monthsButton.setText("X");
158         monthsButton.setFocusable(false);
159         gridConstraints = new GridBagConstraints();
160         gridConstraints.gridx = 2;
161         gridConstraints.gridy = 2;
162         gridConstraints.insets = new Insets(10, 0, 0, 0);
163         getContentPane().add(monthsButton, gridConstraints);
164         monthsButton.addActionListener(new ActionListener()
165         {
166             public void actionPerformed(ActionEvent e)
167             {
168                 monthsButtonActionPerformed(e);
169             }
170         });
171         paymentButton.setText("X");
172         paymentButton.setFocusable(false);
173         gridConstraints = new GridBagConstraints();
174         gridConstraints.gridx = 2;
175         gridConstraints.gridy = 3;
176         gridConstraints.insets = new Insets(10, 0, 0, 0);
```

```
177         getContentPane().add(paymentButton, gridConstraints);
178         paymentButton.addActionListener(new ActionListener()
179         {
180             public void actionPerformed(ActionEvent e)
181             {
182                 paymentButtonActionPerformed(e);
183             }
184         });
185 
186 
187 
188         analysisLabel.setText("Analizando Prestamo:");
189         analysisLabel.setFont(myFont);
190         gridConstraints = new GridBagConstraints();
191         gridConstraints.gridx = 3;
192         gridConstraints.gridy = 0;
193         gridConstraints.anchor = GridBagConstraints.WEST;
194         gridConstraints.insets = new Insets(0, 10, 0, 0);
195         getContentPane().add(analysisLabel, gridConstraints);
196         analysisTextArea.setPreferredSize(new Dimension(250, 150));
197         analysisTextArea.setFocusable(false);
198         analysisTextArea.setBorder
    (BorderFactory.createLineBorder(Color.BLACK));
199         analysisTextArea.setFont(new Font("Nuevo Mensaje", Font.PLAIN,
    14));
```

```
200         analysisTextArea.setEditable(false);
201         analysisTextArea.setBackground(Color.WHITE);
202         gridConstraints = new GridBagConstraints();
203         gridConstraints.gridx = 3;
204         gridConstraints.gridy = 1;
205         gridConstraints.gridheight = 4;
206         gridConstraints.insets = new Insets(0, 10, 0, 10);
207         getContentPane().add(analysisTextArea, gridConstraints);
208         exitButton.setText("Salir");
209         exitButton.setFocusable(false);
210         gridConstraints = new GridBagConstraints();
211         gridConstraints.gridx = 3;
212         gridConstraints.gridy = 5;
213         getContentPane().add(exitButton, gridConstraints);
214         exitButton.addActionListener(new ActionListener(){
215             public void actionPerformed(ActionEvent e)
216             {
217                 exitButtonActionPerformed(e);
218             }
219         });
220         paymentButton.doClick();
221         setSize(700,400);
222     }
223     private void balanceTextFieldActionPerformed(ActionEvent e){
224         balanceTextField.transferFocus();
```

```
225     }
226     private void interestTextFieldActionPerformed(ActionEvent e){
227         interestTextField.transferFocus();
228     }
229     private void monthsTextFieldActionPerformed(ActionEvent e){
230         monthsTextField.transferFocus();
231     }
232     private void paymentTextFieldActionPerformed(ActionEvent e){
233         paymentTextField.transferFocus();
234     }
235     private void exitButtonActionPerformed(ActionEvent e)
236     {
237         System.exit(0);
238     }
239     private void monthsButtonActionPerformed(ActionEvent e)
240     {
241         computePayment = false;
242         paymentButton.setVisible(true);
243         monthsButton.setVisible(false);
244         monthsTextField.setText("");
245         monthsTextField.setEditable(false);
246         monthsTextField.setBackground(lightYellow);
247         paymentTextField.setEditable(true);
248         paymentTextField.setBackground(Color.WHITE);
249         paymentTextField.setFocusable(true);
```

```
250         computeButton.setText("Computarizando Numeros de Pagos");
251         balanceTextField.requestFocus();
252     }
253     private void paymentButtonActionPerformed(ActionEvent e)
254     {
255         computePayment = true;
256         paymentButton.setVisible(false);
257         monthsButton.setVisible(true);
258         monthsTextField.setEditable(true);
259         monthsTextField.setBackground(Color.WHITE);
260         monthsTextField.setFocusable(true);
261         paymentTextField.setText("");
262         paymentTextField.setEditable(false);
263         paymentTextField.setBackground(lightYellow);
264         paymentTextField.setFocusable(false);
265         computeButton.setText("Computarizando Pago Mensual");
266         balanceTextField.requestFocus();
267     }
268     private void computeButtonActionPerformed(ActionEvent e)
269     {
270         double balance, interest, payment=0.0;
271         int months;
272         double monthlyInterest, multiplier;
273         double loanBalance, finalPayment;
274         if(validateDecimalNumber(balanceTextField)) {
```

```
275             balance = Double.valueOf(balanceTextField.getText
    ()).doubleValue();
276         }
277         else{
278             JOptionPane.showConfirmDialog(null, "Entrada de saldo de
    prestamo no valida o vacia.\nPor Favor Corrigalo.", "Error de entrada
    de saldo", JOptionPane.DEFAULT_OPTION,JOptionPane.INFORMATION_MESSAGE);
279             return;
280         }
281         if(validateDecimalNumber(interestTextField)){
282             interest =Double.valueOf(interestTextField.getText
    ()).doubleValue();
283         }
284         else{
285             JOptionPane.showConfirmDialog(null, "Entrada de tasa de
    interes no valida o vacia.\nPor Favor Corrigalo.", "Error de entrada de
    intereses",
    JOptionPane.DEFAULT_OPTION,JOptionPane.INFORMATION_MESSAGE);
286             return;
287         }
288         monthlyInterest = interest / 1200;// Compute loan payment
289         if(computePayment)
290         {
291             if(validateDecimalNumber(monthsTextField))
292                 months = Integer.valueOf(monthsTextField.getText
```

```
   ()).intValue();
293             else{
294                 JOptionPane.showConfirmDialog(null, "Numero de entrada
   de pago no valido o vacio.\nPor Favor Corrigalo.", "Error de entrada
   del número de pagos",JOptionPane.DEFAULT_OPTION,
   JOptionPane.INFORMATION_MESSAGE);
295                 return;
296             }
297             if(interest == 0){
298                 payment = balance / months;
299             }
300             else {
301                 multiplier = Math.pow(1 + monthlyInterest, months);
302                 payment = balance * monthlyInterest * multiplier /
   (multiplier - 1);
303             }
304             paymentTextField.setText(new DecimalFormat("0.00").format
   (payment));
305         }
306         else {
307             if(validateDecimalNumber(paymentTextField))
308                 payment = Double.valueOf(paymentTextField.getText
   ()).doubleValue();
309             if (payment <= (balance * monthlyInterest + 1.0))
310             {
```

```
311                 if (JOptionPane.showConfirmDialog(null, "El pago minimo
    debe ser $" +new DecimalFormat("0.00").format((int)(balance *
    monthlyInterest + 1.0)) + "\n" + "¿Quieres utilizar el pago minimo?",
    "Error de Entrada",
    JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE) ==
    JOptionPane.YES_OPTION)
312                 {
313                     paymentTextField.setText(new DecimalFormat
    ("0.00").format((int)(balance * monthlyInterest + 1.0)));
314                     payment =Double.valueOf(paymentTextField.getText
    ()).doubleValue();
315                 }
316                 else
317                     {
318                         paymentTextField.requestFocus();return;
319                     }
320             }
321             else {
322                 JOptionPane.showConfirmDialog(null, "Entrada de pago
    mensual no valida o vacia.\nPor Favor de Corregirlo.", "Error de
    Entrada de Pago",
    JOptionPane.DEFAULT_OPTION,JOptionPane.INFORMATION_MESSAGE);
323                 return;
324             }
325             if(interest == 0){
```

```
326                 months = (int)(balance / payment);
327             }
328             else {
329                 months = (int) ((Math.log(payment) - Math.log(payment -
    balance * monthlyInterest)) / Math.log(1 + monthlyInterest));
330             }
331                 monthsTextField.setText(String.valueOf(months));
332         }
333         payment =Double.valueOf(paymentTextField.getText()).doubleValue
    ();
334         analysisTextArea.setText("Balance del Prestamo: $" + new
    DecimalFormat("0.00").format(balance));
335         analysisTextArea.append("\n" + "Tasa de Interes: " + new
    DecimalFormat("0.00").format(interest) + "%");// process all but last
    payment
336         loanBalance = balance;
337         for (int paymentNumber = 1; paymentNumber <= months - 1;
    paymentNumber++)
338         {
339             loanBalance += loanBalance * monthlyInterest - payment;
340         }// find final payment
341         finalPayment = loanBalance;
342         if (finalPayment > payment){// apply one more payment
343             loanBalance += loanBalance * monthlyInterest - payment;
344             finalPayment = loanBalance;months++;
```

```
345             monthsTextField.setText(String.valueOf(months));
346         }
347         analysisTextArea.append("\n\n" + String.valueOf(months - 1) + "
    Pagos de $" + new DecimalFormat("0.00").format(payment));
348         analysisTextArea.append("\n" + "Pago Final de: $" + new
    DecimalFormat("0.00").format(finalPayment));
349         analysisTextArea.append("\n" + "Total de Pagos: $" + new
    DecimalFormat("0.00").format((months - 1) * payment + finalPayment));
350         analysisTextArea.append("\n" + "Interes de Pago $" + new
    DecimalFormat("0.00").format((months - 1) * payment + finalPayment -
    balance));
351         computeButton.setEnabled(false);
352         newLoanButton.setEnabled(true);
353         newLoanButton.requestFocus();
354     }
355     private void newLoanButtonActionPerformed(ActionEvent e)
356     {
357         if(computePayment){
358             paymentTextField.setText("");
359         }
360         else{
361             monthsTextField.setText("");
362         }
363         analysisTextArea.setText("");
364         computeButton.setEnabled(true);
```

```
365         newLoanButton.setEnabled(false);
366         balanceTextField.requestFocus();
367     }
368     private void exitForm(WindowEvent evt){
369         System.exit(0);
370     }
371     public boolean validateDecimalNumber(JTextField tf){
372         String s = tf.getText().trim();
373         boolean hasDecimal = false;
374         boolean valid = true;
375         if(s.length() == 0){
376             valid = false;
377         }
378         else{
379             for(int i=0;i<s.length();i++){
380                 char c = s.charAt(i);
381                 if(c >= '0' && c <= '9'){
382                     continue;
383                 }
384                 else if(c == '.' && !hasDecimal){
385                     hasDecimal = true;
386                 }
387                 else{
388                     valid = false;
389                 }
```

```
390             }
391         }
392         tf.setText(s);
393         if(!valid){
394             tf.requestFocus();
395         }
396         return(valid);
397 
398 
399 }
400 }
```