

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA  
FACULTAD DE INGENIERIA  
ESCUELA DE CIENCIAS Y SISTEMAS  
INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1  
SECCIÓN A  
SEGUNDO SEMESTRE 2022  
ING. MARLON FRANCISCO ORELLANA LOPEZ  
AUX. DIEGO ALEJANDRO VASQUEZ



BRANDON EDUARDO PABLO GARCIA  
202112092  
Guatemala, septiembre del 2022

## CONTENIDO

Introducción.....	1
Objetivos.....	2
Contenido Técnico.....	3
Inicio.....	3
Creación de usuario.....	5
Creación de cuentas.....	10
Ver Cuentas.....	14
Operaciones.....	17



## **INTRODUCCION**

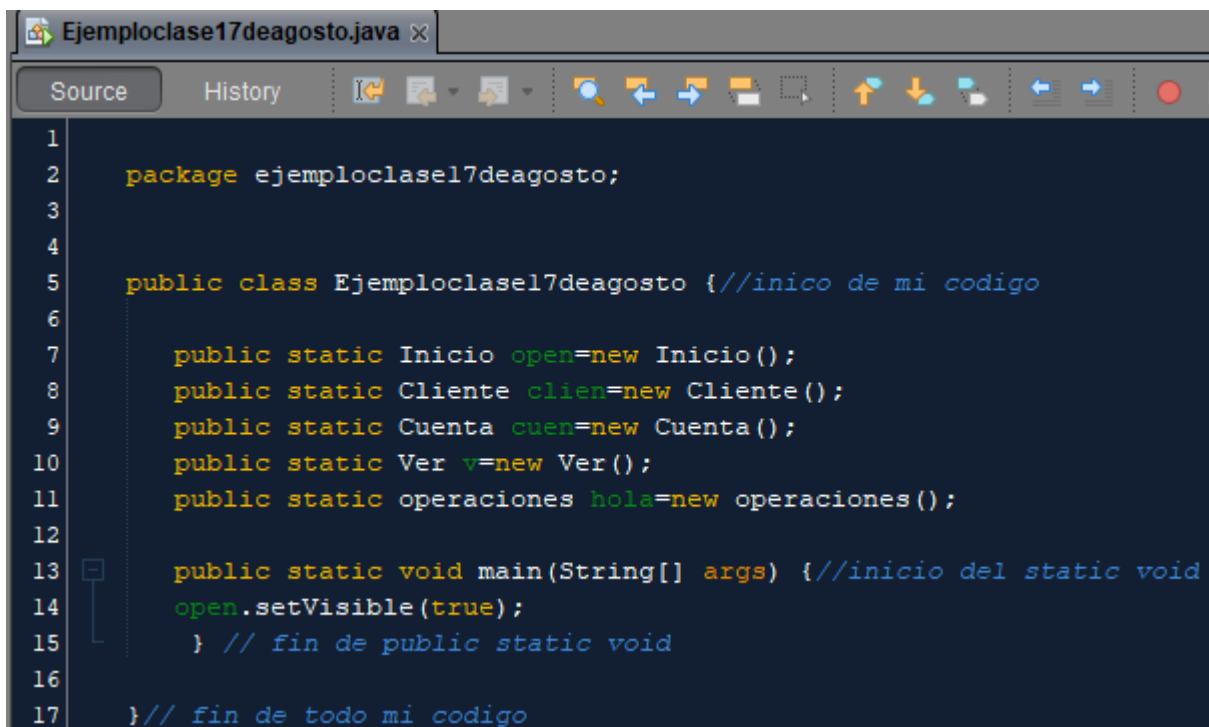
Este manual describe los pasos necesarios para cualquier persona que tenga ciertas bases de sistemas pueda realizar el código implementado en Java NetBens donde se crea un código para un sistema bancario utilizando POO (Programación Orientada a Objetos) y así poder implementarlo de la mejor manera. El siguiente código se explicó de la manera mas detalla posible para la mejor comprensión de la persona.

## **OBJETIVOS**

- Brindar la información necesaria para poder representar la funcionalidad técnica de la estructura, diseño y definición del aplicativo.
- Describir las herramientas utilizadas para el diseño y desarrollo del prototipo

## CONTENIDO TECNICO

Comienza en la clase principal en donde se derivarán los demás JFrames que usaremos cada uno con su nombre y su variable que lo identifica, como lo son el Inicio, la creación de los clientes denominada “Clientes”, las cuentas, la parte donde se miran las cuentas ya creadas con sus números de identificación denominada “Ver” y por ultimo las operaciones donde se mostraran todo lo que conlleva las operaciones bancarias y el historial de transacciones.



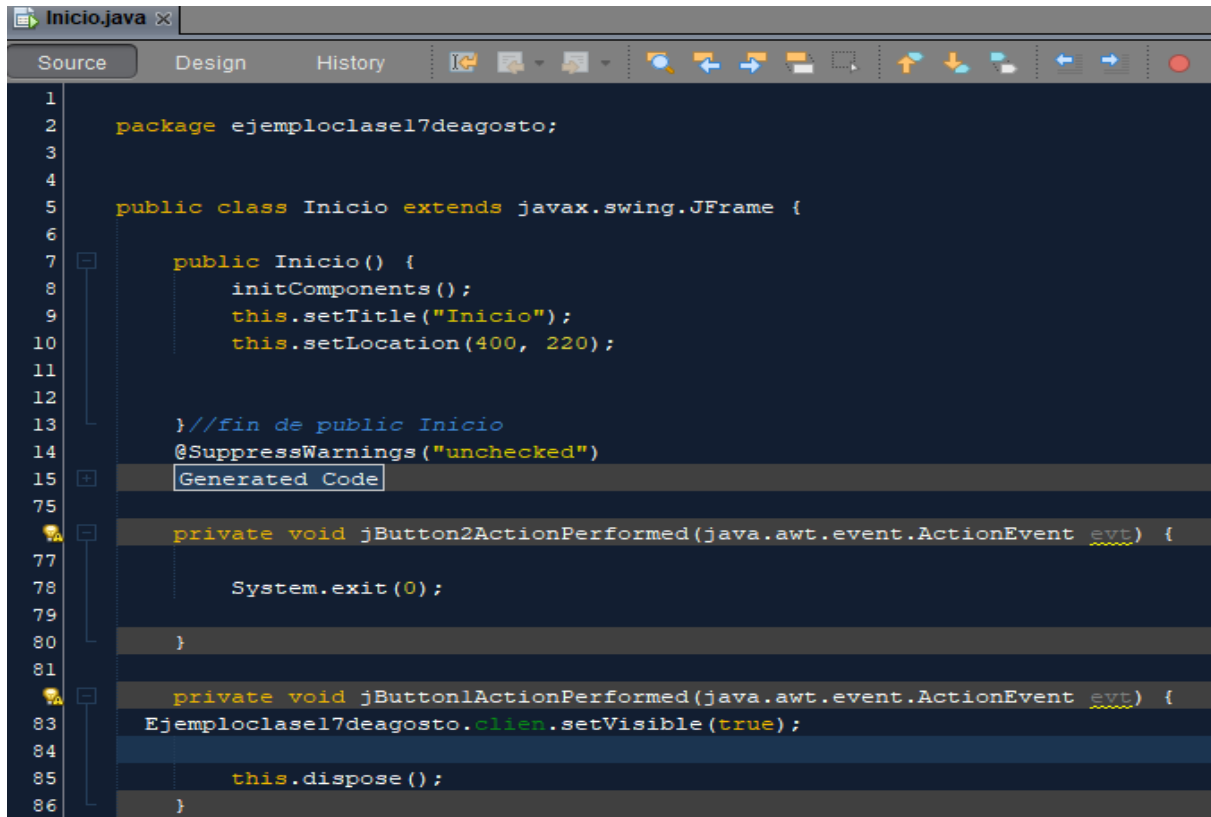
```
1
2 package ejemploclase17deagosto;
3
4
5 public class Ejemploclase17deagosto {//inicio de mi codigo
6
7     public static Inicio open=new Inicio();
8     public static Cliente clien=new Cliente();
9     public static Cuenta cuen=new Cuenta();
10    public static Ver v=new Ver();
11    public static operaciones hola=new operaciones();
12
13    public static void main(String[] args) {//inicio del static void
14        open.setVisible(true);
15    } // fin de public static void
16
17 }// fin de todo mi codigo
```

### Inicio:

En la interface denominada inicio se utilizó el método java.swing.JFrame para que de una manera más optima se realizara lo antes mencionado. También se aplicaron otros métodos como el set que nos ayudaron a colocar titulo a la ventana emergente “setTitle” y a la ubicación de la pantalla “setLocation.

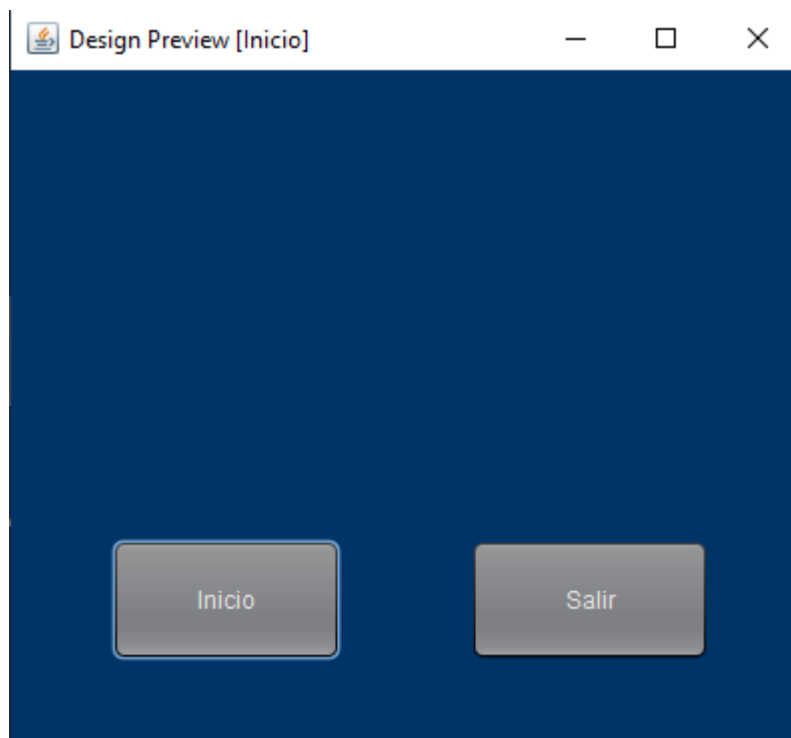
También se utilizaron dos Buttons, uno para iniciar el programa el cual nos mandara a la ventana siguiente, esto fue posible utilizando el método setVisible donde

colocamos el nombre de la ventana a donde nos queremos dirigir, cuidado que tiene que ser el nombre de su variable. Y el otro Button se utilizó el método `System.exit(0)` que hace que el programa finalice por completo.



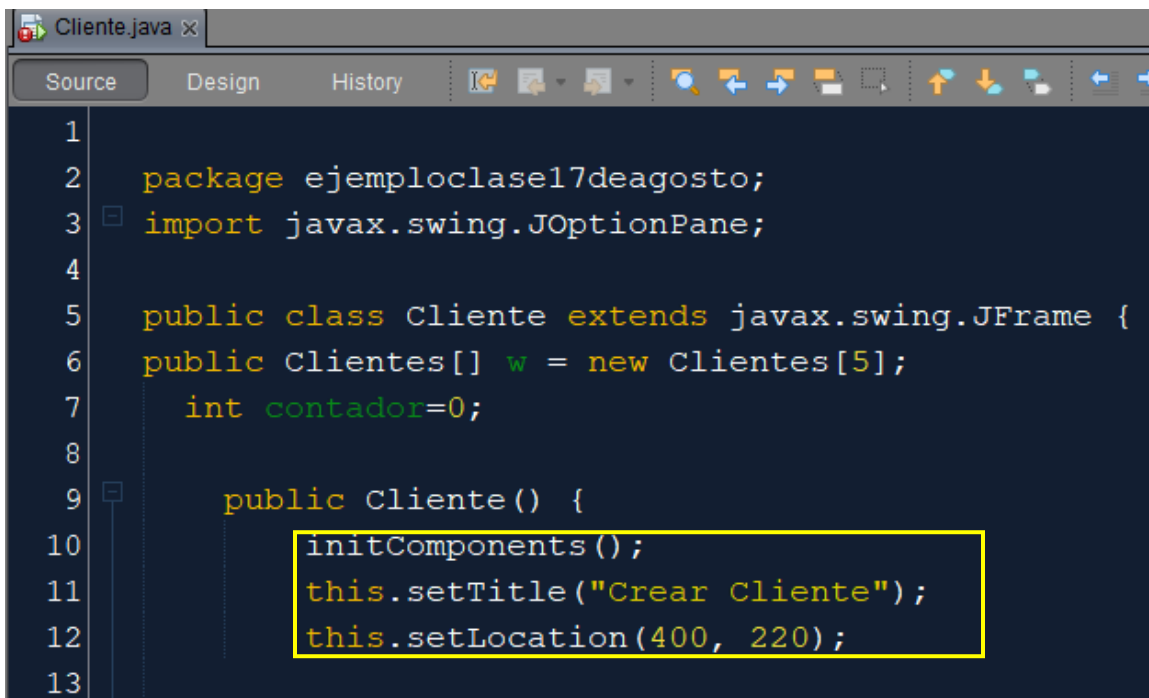
```
1 package ejemploclase17deagosto;
2
3
4
5 public class Inicio extends javax.swing.JFrame {
6
7     public Inicio() {
8         initComponents();
9         this.setTitle("Inicio");
10        this.setLocation(400, 220);
11
12    } //fin de public Inicio
13    @SuppressWarnings("unchecked")
14    Generated Code
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
78     System.exit(0);
79 }
80
81
82
83 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
84     Ejemploclase17deagosto.clien.setVisible(true);
85     this.dispose();
86 }
```

La interfaz que vera el usuario será la siguiente.



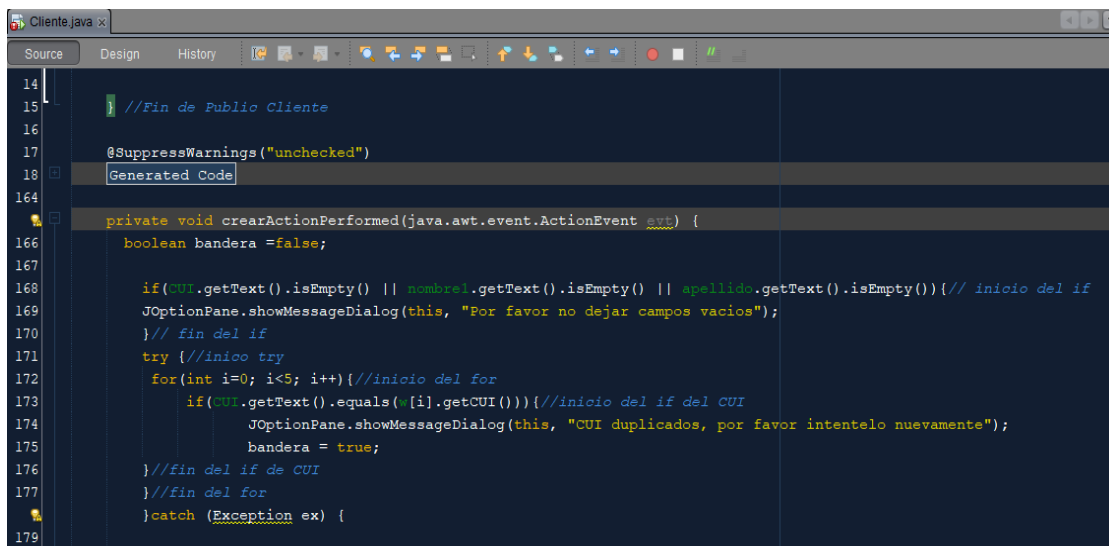
## Creación de usuario:

En el código que se usó para el registro de los clientes para comenzar se utilizó el método set para el título de la ventana emergente con “setTitle” y la ubicación del mismo con “setLocation”.



```
1
2 package ejemploclase17deagosto;
3 import javax.swing.JOptionPane;
4
5 public class Cliente extends javax.swing.JFrame {
6     public Clientes[] w = new Clientes[5];
7     int contador=0;
8
9     public Cliente() {
10         initComponents();
11         this.setTitle("Crear Cliente");
12         this.setLocation(400, 220);
13     }
```

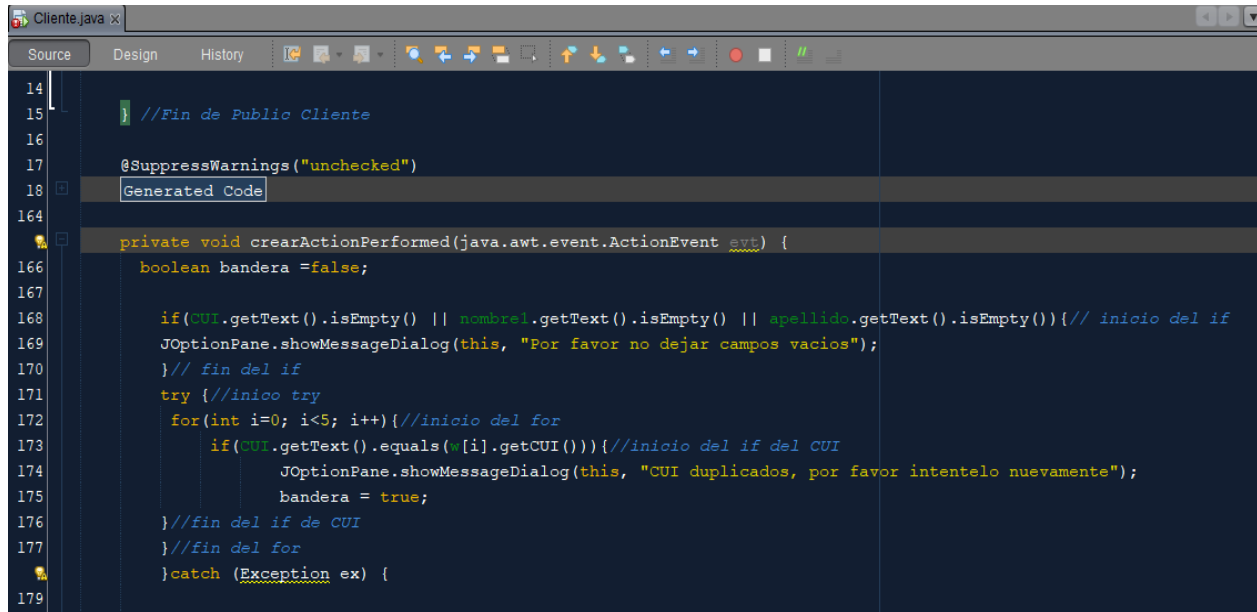
Después de eso se utilizó el método if para verificar si en cada cuadro de texto no se dejaron espacio en blanco utilizando el método de `getTexts().isEmpty()`, ya que si ese fuera el caso de mandar un mensaje emergente con la función `JOptionPane` para el mismo se tuvo que exportar la librería `javax.swing.JOptionPane`.



```
14
15 //Fin de Public Cliente
16
17 @SuppressWarnings("unchecked")
18 Generated Code
19
164
165 private void crearActionPerformed(java.awt.event.ActionEvent evt) {
166     boolean bandera = false;
167
168     if(CUI.getText().isEmpty() || nombre.getText().isEmpty() || apellido.getText().isEmpty()){// inicio del if
169         JOptionPane.showMessageDialog(this, "Por favor no dejar campos vacios");
170     }// fin del if
171     try { //inicio try
172         for(int i=0; i<5; i++){ //inicio del for
173             if(CUI.getText().equals(w[i].getCUI())){//inicio del if del CUI
174                 JOptionPane.showMessageDialog(this, "CUI duplicados, por favor intentelo nuevamente");
175                 bandera = true;
176             } //fin del if de CUI
177         } //fin del for
178     } catch (Exception ex) {
179     }
```



Se tuvo que utilizar un for para ir contando cuantos usuarios se están creando, por lo mismo se tuvo que declarar una variable en booleana y dentro del mismo se creó un if para verificar si el CUI del usuario es duplicado, si ese fuera el caso se mandara un mensaje con la función JOptionPane.



```
14
15 //Fin de Public Cliente
16
17 @SuppressWarnings("unchecked")
18 Generated Code
164
165 private void crearActionPerformed(java.awt.event.ActionEvent evt) {
166     boolean bandera = false;
167
168     if(CUI.getText().isEmpty() || nombre1.getText().isEmpty() || apellido.getText().isEmpty()){// inicio del if
169         JOptionPane.showMessageDialog(this, "Por favor no dejar campos vacios");
170     }// fin del if
171     try { //inicio try
172         for(int i=0; i<5; i++){//inicio del for
173             if(CUI.getText().equals(w[i].getCUI())){//inicio del if del CUI
174                 JOptionPane.showMessageDialog(this, "CUI duplicados, por favor intentelo nuevamente");
175                 bandera = true;
176             }//fin del if de CUI
177         }//fin del for
178     }catch (Exception ex) {
179
```

Al terminar todo eso se declara una variable llamada “contador” y la igualamos a cero para que comience a contar desde ahí, también declaramos el array para clientes que tendrá un máximo de cinco clientes. Dentro de if de contador colocamos todo lo que se necesita para el ingreso de clientes con el método getText para ir ingresando y creando cada cliente donde mostrara un mensaje con un JOptionPane que dira que el cliente fue creado exitosamente.



```
180
181
182 if(contador==5){
183     JOptionPane.showMessageDialog(this, "No es posible crear mas usuarios");
184     Ejemploclase17deagosto.cuen.setVisible(true);
185
186     this.setVisible(false);
187
188 }//fin del if cui 5
189 else if(bandera==false){//inicio del else if
190     w[contador] = new Clientes (CUI.getText(), nombre1.getText(), apellido.getText());
191     System.out.println(w[contador].getCUI()+w[contador].getnombre()+w[contador].getapellido());
192     contador = contador+1;
193     JOptionPane.showMessageDialog(this, "Cliente creado exitosamente");
194 }//fin de else if
195
```

Siguiente de eso programamos los tres TextField usando la función “char” y un if para que los TextField del CUI solo acepte número, el de los TextField de nombres y apellidos para que solo acepte letras.

```

199
200 private void CUIKeyTyped(java.awt.event.KeyEvent evt) {
201     char x = evt.getKeyChar();
202     if(x<'0' || x>'9') evt.consume();
203 }
204
205 private void nombre1KeyTyped(java.awt.event.KeyEvent evt) {
206     char x = evt.getKeyChar();
207     if((x<'a' || x>'z') && (x<'A' || x>'Z') && (x<' ' || x>' ')) evt.consume();
208 }
209

```

Por último se colocó un Button usando el método setVisible(true) y this.dispose(); que nos ayudara para continuar al siguiente JFrame por si el usuario no desea crear los cinco clientes pueda continuar sin ningún problema.

```

209
210 private void ContinuarActionPerformed(java.awt.event.ActionEvent evt) {
211     Ejemploclase17deagosto.cuen.setVisible(true);
212
213     this.dispose();
214 }
215

```

Para terminar con el código de la siguiente manera.

```

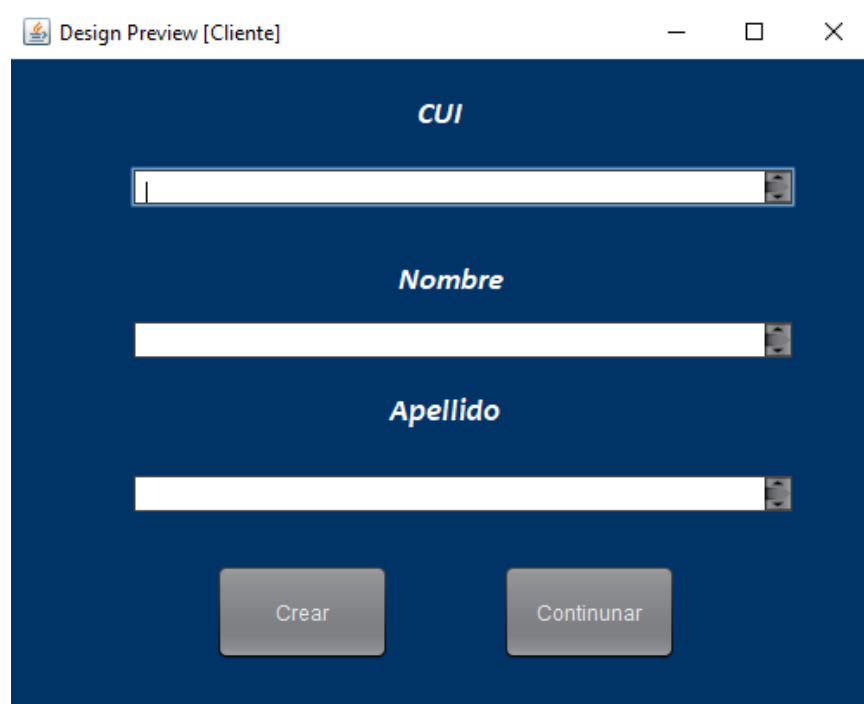
Cliente.java
Source Design History
1
2 package ejemploclase17deagosto;
3 import javax.swing.JOptionPane;
4
5 public class Cliente extends javax.swing.JFrame {
6     public Clientes[] w = new Clientes[5];
7     int contador=0;
8
9     public Cliente() {
10         initComponents();
11         this.setTitle("Crear Cliente");
12         this.setLocation(400, 220);
13
14     } //Fin de Public Cliente
15
16     @SuppressWarnings("unchecked")
17     Generated Code
18
164
165 private void crearActionPerformed(java.awt.event.ActionEvent evt) {
166     boolean bandera =false;
167
168     if(CUI.getText().isEmpty() || nombre1.getText().isEmpty() || apellido.getText().isEmpty()){// inicio del if
169         JOptionPane.showMessageDialog(this, "Por favor no dejar campos vacios");
170     }// fin del if
171     try { //inicio try
172         for(int i=0; i<5; i++){//inicio del for

```

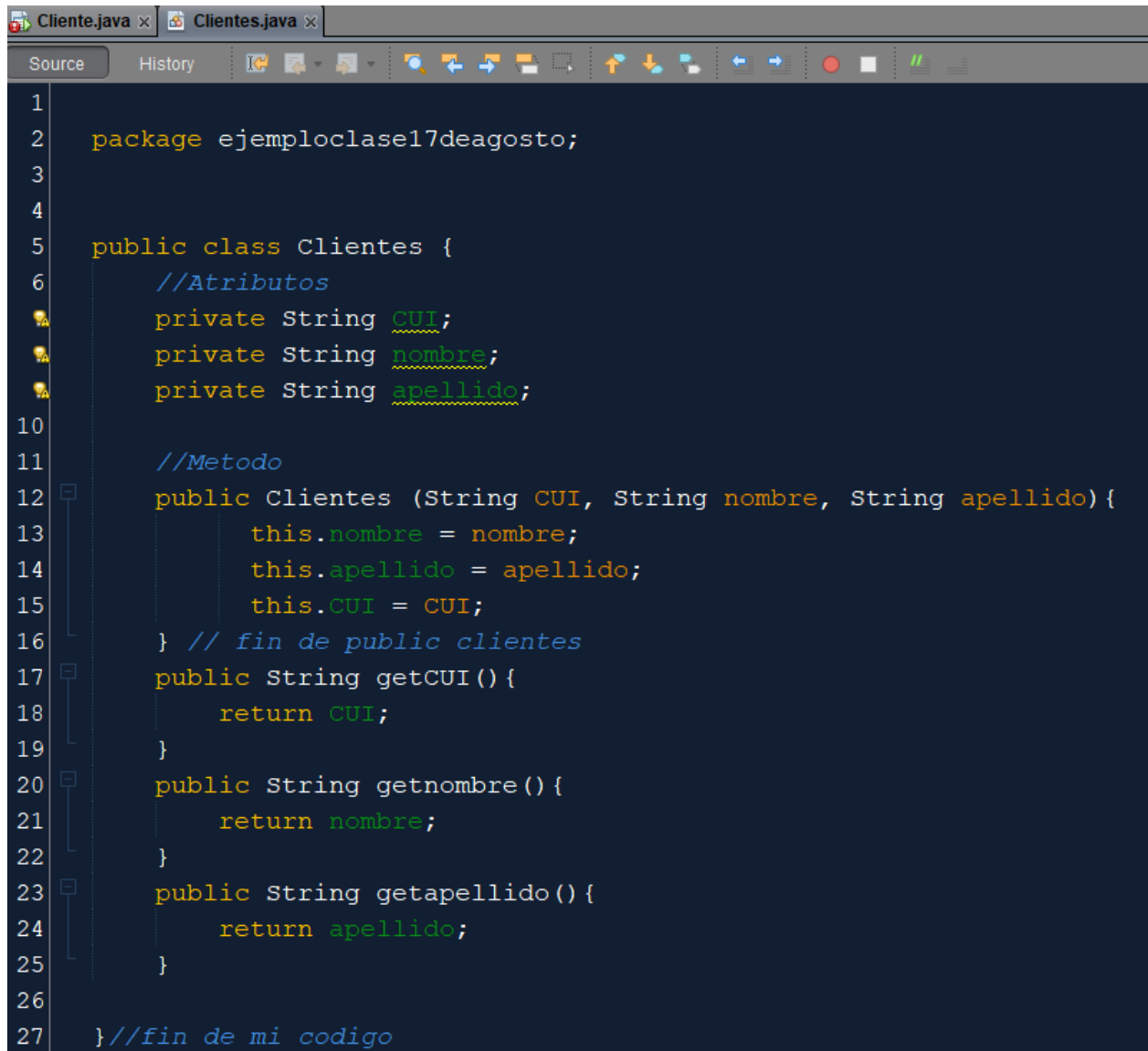
```
Cliente.java x
Source Design History
173         if(CUI.getText().equals(w[i].getCUI())){//inicio del if del CUI
174             JOptionPane.showMessageDialog(this, "CUI duplicados, por favor intentelo nuevamente");
175             bandera = true;
176         }//fin del if de CUI
177     }//fin del for
178     }catch (Exception ex) {
179
180     }
181
182     if(contador==5){
183         JOptionPane.showMessageDialog(this, "No es posible crear mas usuarios");
184         Ejemploclase17deagosto.cuen.setVisible(true);
185
186         this.setVisible(false);
187
188     }//fin del if cui 5
189     else if(bandera==false){//inicio del else if
190         w[contador] = new Clientes (CUI.getText(), nombre1.getText(), apellido.getText());
191         System.out.println(w[contador].getCUI()+" "+w[contador].getnombre()+w[contador].getapellido());
192         contador = contador+1;
193         JOptionPane.showMessageDialog(this, "Cliente creado exitosamente");
194     }//fin de else if
195
196
197
198     }
199
200
201 private void CUIKeyTyped(java.awt.event.KeyEvent evt) {
202     char x = evt.getKeyChar();
203     if(x<'0' || x>'9') evt.consume();
204 }
205
206 private void nombre1KeyTyped(java.awt.event.KeyEvent evt) {
207     char x = evt.getKeyChar();
208     if((x<'a' || x>'z') && (x<'A' || x>'Z') && (x<' ' || x>' ')) evt.consume();
209 }
210
211 private void ContinuarActionPerformed(java.awt.event.ActionEvent evt) {
212     Ejemploclase17deagosto.cuen.setVisible(true);
213     this.dispose();
214 }
215
216 /**
217  * @param args the command line arguments
218  */
219 public static void main(String args[]) {
220     /* Set the Nimbus look and feel */
221     Look and feel setting code (optional)
222
223     /* Create and display the form */
224     java.awt.EventQueue.invokeLater(new Runnable() {
225         public void run() {
```

```
Cliente.java
Source Design History
220 /* Set the Nimbus look and feel */
221 Look and feel setting code (optional)
242
243 /* Create and display the form */
244 java.awt.EventQueue.invokeLater(new Runnable() {
245     public void run() {
246         new Cliente ().setVisible(true);
247     }
248 });
249 }
250
251 // Variables declaration - do not modify
252 public static javax.swing.JTextArea CUI;
253 private javax.swing.JButton Continuar;
254 private javax.swing.JTextArea apellido;
255 private javax.swing.JButton crear;
256 private javax.swing.JLabel jLabel1;
257 private javax.swing.JLabel jLabel2;
258 private javax.swing.JLabel jLabel3;
259 private javax.swing.JPanel jPanel1;
260 private javax.swing.JScrollPane jScrollPane1;
261 private javax.swing.JScrollPane jScrollPane3;
262 private javax.swing.JScrollPane nombre;
263 private javax.swing.JTextArea nombre1;
264 // End of variables declaration
265 } //Fin del Form
266
```

La interfaz gráfica se vería de la siguiente manera.



Para poder crear cada cliente se tuvo que utilizar la programación orientada a objetos por lo cual tuvimos que crear un Publica class denominada como “Clientes” y en ella crear cada cliente usando el motodo String y de nuevamente usar el método get y this para así llamar a cada variable.

A screenshot of a Java IDE window with two tabs: 'Cliente.java' and 'Clientes.java'. The 'Clientes.java' tab is active, showing a Java class named 'Clientes'. The code is as follows:

```
1 package ejemploclase17deagosto;
2
3
4
5 public class Clientes {
6     //Atributos
7     private String CUI;
8     private String nombre;
9     private String apellido;
10
11     //Metodo
12     public Clientes (String CUI, String nombre, String apellido){
13         this.nombre = nombre;
14         this.apellido = apellido;
15         this.CUI = CUI;
16     } // fin de public clientes
17     public String getCUI(){
18         return CUI;
19     }
20     public String getnombre(){
21         return nombre;
22     }
23     public String getapellido(){
24         return apellido;
25     }
26
27 } //fin de mi codigo
```

### Creación de cuentas:

De la misma manera que los de más JFrame aquí utilizamos los métodos setTitle(); y setLocation(); para poder colocarle titulo a la ventana emergente y la ubicación dentro del pantalla. También se llamó a la librería javax.swing.JOptionPane; método que usaremos mas adelante a la misma vez se declararon dos variable usando “int” para tener un contador de cuentas que puede crear cada usuario el otro que será el

ID que tendrá cada cuenta. Se usaron dos arrays el de clientes y el de cuentas denominada "Cuen" con variable "f" que se usara para que cada cliente tenga cinco cuentas cada uno un total de veinticinco cuentas.

```
Cuenta.java x
Source Design History
1 package ejemploclase17deagosto;
2
3 import javax.swing.JOptionPane;
4
5
6 public class Cuenta extends javax.swing.JFrame {
7     public Clientes[] fabiola= new Clientes[5];
8     public Cuen[] f= new Cuen[25];
9     int conta=0;
10    int ID=1439;
11    public Cuenta() {
12        initComponents();
13        this.setTitle("Cuenta");
14        this.setLocation(400, 220);
15
16
17    } //fin de public cuenta
```

Con la utilizaci3n de un Button y tambi3n los metodos get haremos posible la creaci3n de cada una de las cuentas que se mostraran en el ComboBox claramente teniendo un orden empezando por el CUI seguido por el nombre y el apellido. Tambien se implento el uso del JOptionPane para mostrar un mensaje diciendo "Cuenta creada exitosamente"

Tambi3n declaramos un if con la varibale "conta" igualada a veinticinco que se usa para saber cuantas cuentas han sido creadas para cada cliente.

```
Cuenta.java x
Source Design History
private void CrearActionPerformed(java.awt.event.ActionEvent evt) {
105    f[conta] = new Cuen (ID, fabiola[barrita.getSelectedIndex()],0);
106    System.out.println(f[conta].getID()+f[conta].getcliente().getnombre()+f[conta].getsaldo());
107    conta = conta+1;
108    ID = ID+1;
109    JOptionPane.showMessageDialog(this, "Cuenta creado exitosamente");
110    if(conta==25){
111        JOptionPane.showMessageDialog(this, "No es posible crear mas usuarios");
112        Ejemploclase17deagosto.v.setVisible(true);
113
114        this.setVisible(false);
115
116    } //fin del if
```

Continuamente de eso usamos una de las propiedades del panel en la parte de eventos que nos abrirá una opción llamada "formWindowOpened" la cual nos ayudará a ingresar la información mandada del anterior JFrame llamada Clientes y así ingresarla en el ComboBox llamada "barrita". Claramente también se necesito el uso de get y addItem.

Con el uso más eficiente se colocó otro Button llamado "continuar" que nos ayudara a pasar al otro JFrame denominada como "Ver".

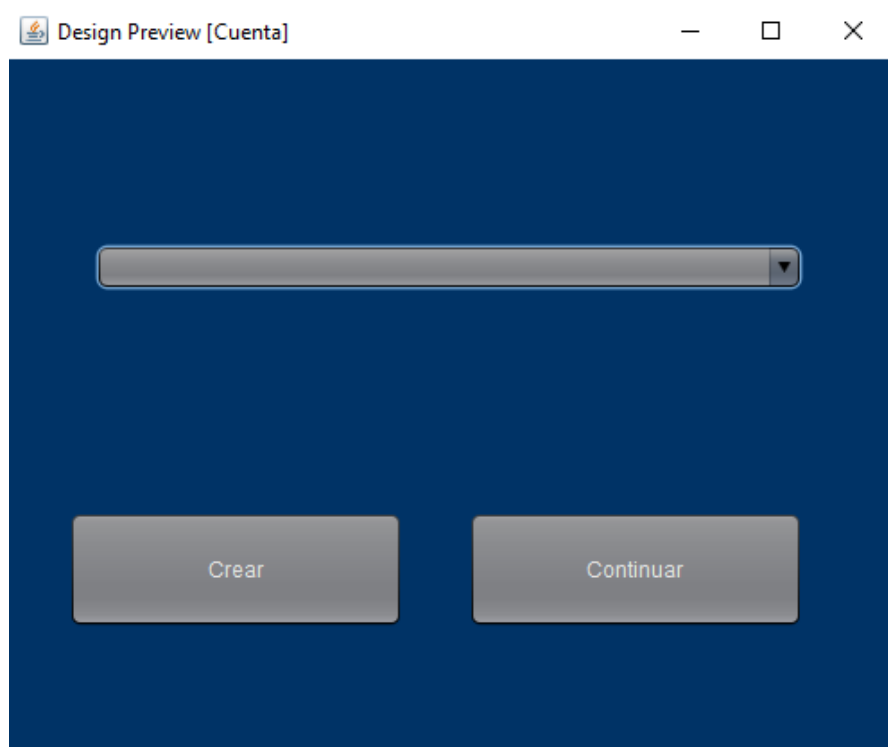
```
Cuenta.java x
Source Design History
122
123     }
124
125     private void formWindowOpened(java.awt.event.WindowEvent evt) {
126         for(int i=0;i<5;i++){//inicio del for
127             fabiola[i]= Ejemploclase17deagosto.clien.w[i];
128             barrita.addItem(fabiola[i].getCUI()+" "+fabiola[i].getnombre()+" "+fabiola[i].getapellido());
129         }//fin del for
130     }
131
132     private void CrearActionPerformed(java.awt.event.ActionEvent evt) {
133         Ejemploclase17deagosto.v.setVisible(true);
134
135         this.dispose();
136     }
```

Finalizando esta parte de cuentas con el código de la siguiente manera:

```
Cuenta.java x
Source Design History
1 package ejemploclase17deagosto;
2
3 import javax.swing.JOptionPane;
4
5
6 public class Cuenta extends javax.swing.JFrame {
7     public Clientes[] fabiola= new Clientes[5];
8     public Cuen[] f= new Cuen[25];
9     int conta=0;
10    int IF=1439;
11
12    public Cuenta() {
13        initComponents();
14        this.setTitle("Cuenta");
15        this.setLocation(400, 220);
16
17    }//fin de public cuenta
18
19    Generated Code
20
21    private void barritaActionPerformed(java.awt.event.ActionEvent evt) {
22
23    }
24
25    private void CrearActionPerformed(java.awt.event.ActionEvent evt) {
26        f[conta] = new Cuen (ID, fabiola[barrita.getSelectedIndex()],0);
27        System.out.println(f[conta].getID()+f[conta].getcliente().getnombre()+f[conta].getsaldo());
28        conta = conta+1;
29        ID = IF+1;
30        JOptionPane.showMessageDialog(this, "Cuenta creado exitosamente");
31        if (conta==25){
32            JOptionPane.showMessageDialog(this, "No es posible crear mas usuarios");
33            Ejemploclase17deagosto.v.setVisible(true);
34
35            this.setVisible(false);
36        }
37    }//fin del if
38 }
```

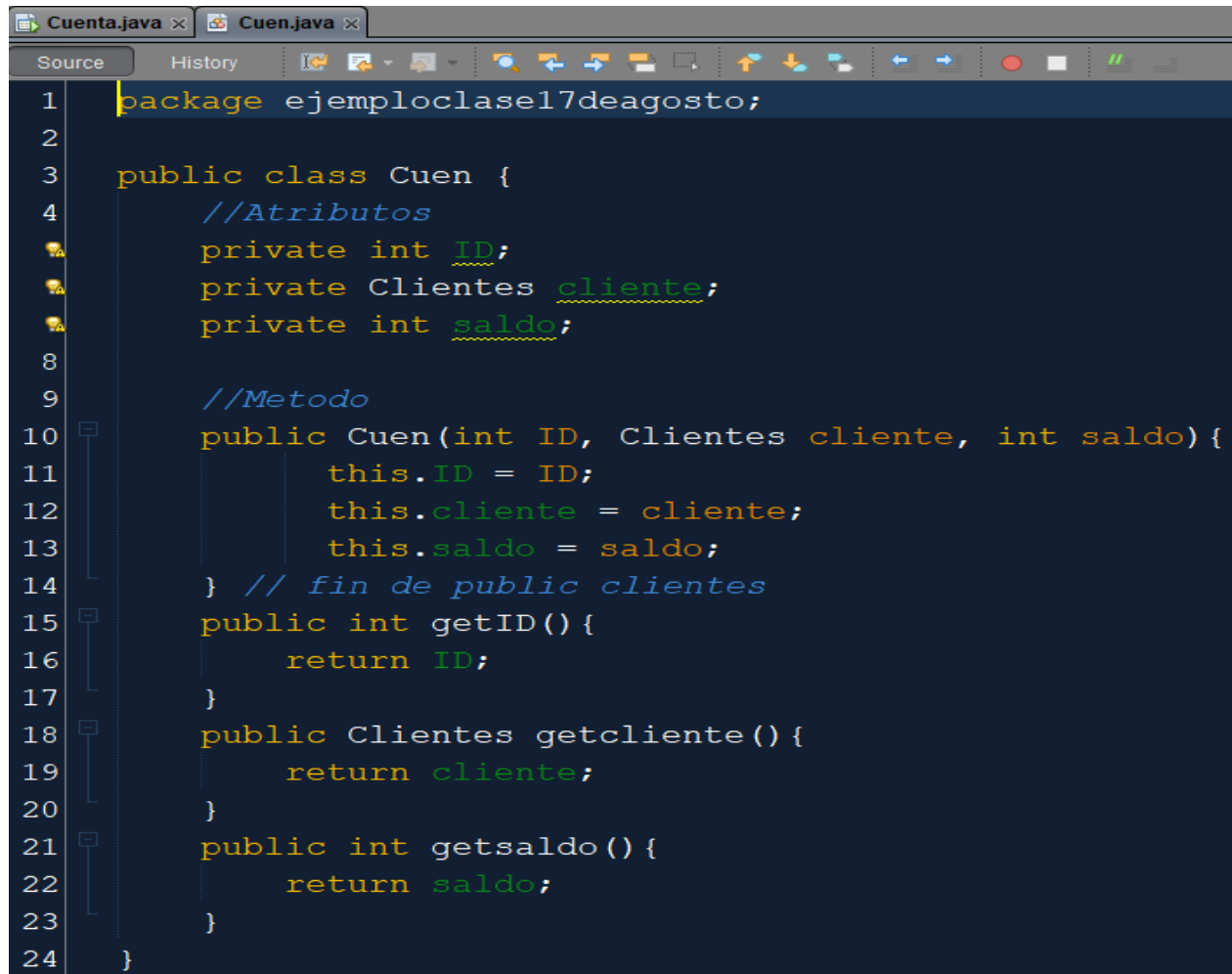
```
Cuenta.java x
Source Design History
122 }
123
124
125 private void formWindowOpened(java.awt.event.WindowEvent evt) {
126     for(int i=0;i<5;i++){//inicio del for
127         fabiola[i]= Ejemploclase17deagosto.clien.w[i];
128         barra1.addItem(fabiola[i].getCUI()+" "+fabiola[i].getnombre()+" "+fabiola[i].getapellido());
129     }//fin del for
130 }
131
132 private void CrearActionPerformed(java.awt.event.ActionEvent evt) {
133     Ejemploclase17deagosto.v.setVisible(true);
134
135     this.dispose();
136 }
137
138 /**
139  * @param args the command line arguments
140  */
141 public static void main(String args[]) {
142     /* Set the Nimbus look and feel */
143     Look and feel setting code (optional)
144
145     /* Create and display the form */
146     java.awt.EventQueue.invokeLater(new Runnable() {
147         public void run() {
148             new Cuenta().setVisible(true);
149         }
150     });
151 }
152
153 // Variables declaration - do not modify
154 private javax.swing.JButton crear;
155 private javax.swing.JButton Crear1;
156 public static javax.swing.JComboBox<String> barra1;
157 private javax.swing.JPanel jPanel1;
158 // End of variables declaration
159 }
```

Y la interfaz gráfica de la siguiente manera:





Aquí también nos fue útil la programación orientada a objetos donde declaramos una public class llamada “Cuen” y las variables usando el método “int” exceptuando la parte del cliente que esa parte saldrá del objeto Clientes. También se implemento el uso de los get que nos ayuda mandar y obtener la información que se ingresa.

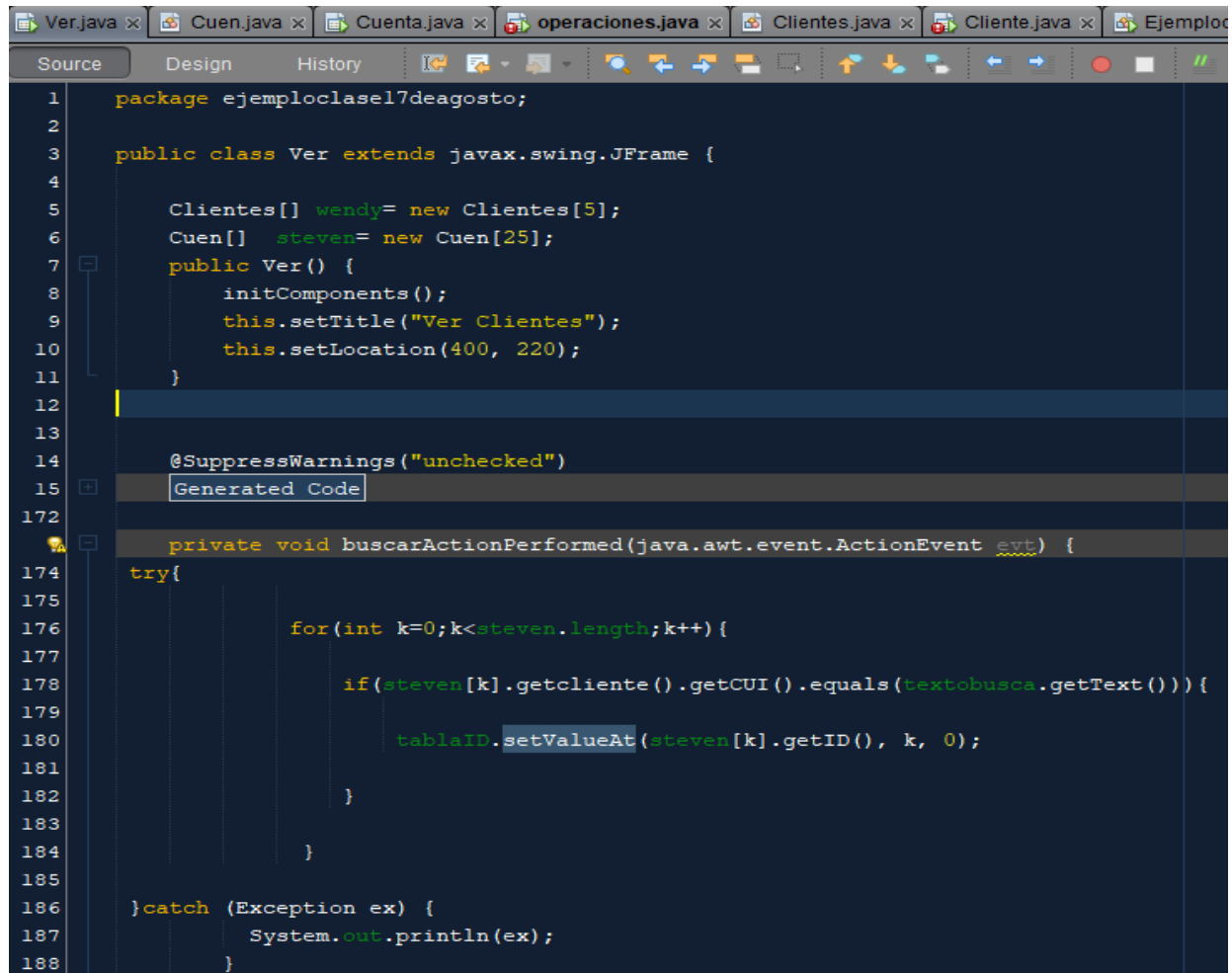


```
1 package ejemploclase17deagosto;
2
3 public class Cuen {
4     //Atributos
5     private int ID;
6     private Clientes cliente;
7     private int saldo;
8
9     //Metodo
10    public Cuen(int ID, Clientes cliente, int saldo){
11        this.ID = ID;
12        this.cliente = cliente;
13        this.saldo = saldo;
14    } // fin de public clientes
15    public int getID(){
16        return ID;
17    }
18    public Clientes getcliente(){
19        return cliente;
20    }
21    public int getsaldo(){
22        return saldo;
23    }
24 }
```

### Ver cuentas:

De la misma manera que los de más JFrame aquí utilizamos los métodos setTitle(); y setLocation(); para poder colocarle título a la ventana emergente y la ubicación dentro del pantalla. Se implemento de los arrays Clientes con nombre de variable “wendy” con límite de cinco que es para la creación de cada cliente y Cuenta con nombre de variable “steven” con límite de veinticinco que es para las cuentas.

Para poder encontrar los números de cuentas asociadas usamos el Button denominado “buscar” con un for y un if que nos ayudara a buscar las cuentas a través del TextFile cuando el usuario coloque su CUI para eso necesitamos la ayuda de los get y mostrarlo en la tabla con nombre “tablaID”.



```

1 package ejemploclase17deagosto;
2
3 public class Ver extends javax.swing.JFrame {
4
5     Clientes[] wendy= new Clientes[5];
6     Cuen[] steven= new Cuen[25];
7     public Ver() {
8         initComponents();
9         this.setTitle("Ver Clientes");
10        this.setLocation(400, 220);
11    }
12
13
14    @SuppressWarnings("unchecked")
15    Generated Code
16
172 private void buscarActionPerformed(java.awt.event.ActionEvent evt) {
174     try{
175
176         for(int k=0;k<steven.length;k++){
177
178             if(steven[k].getcliente().getCUI().equals(textobusca.getText())){
179
180                 tablaID.setValueAt(steven[k].getID(), k, 0);
181
182             }
183
184         }
185
186     }catch (Exception ex) {
187         System.out.println(ex);
188     }
189 }

```

Continuamente utilizamos un evento del JFrame Ver para así poder trasladar la información de los usuarios creados y colocarlo en la tabla principal donde aparecerá su número de CUI, nombre y apellido. Para poder lograr esto se utilizó un for y adentro declaramos nuestra variable cuenta llamada Steven y la de clientes llamada Wendy ambos que comenzaran en la posición “i”

Por ende, cada cliente creado se colocará en la tabla llamada tabla1 con su orden respectivo.

```

188 }
189
190 }
191
192 private void formWindowOpened(java.awt.event.WindowEvent evt) {
193     try{
194         for(int i=0;i<5;i++){//inicio del for
195             wendy[i]= Ejemploclase17deagosto.clien.w[i];
196             tabla1.SetValueAt(wendy[i].getCUI(), i, 0);
197             tabla1.SetValueAt(wendy[i].getnombre(), i, 1);
198             tabla1.SetValueAt(wendy[i].getapellido(), i, 2);
199         }//fin del for
200     }catch (Exception ex) {
201
202     }
203     try{
204         for(int i=0;i<steven.length;i++){//inicio del for2
205             steven[i]= Ejemploclase17deagosto.cuen.f[i];
206         }//fin del for2
207     }catch (Exception ex) {
208
209     }
210
211
212
213 }

```

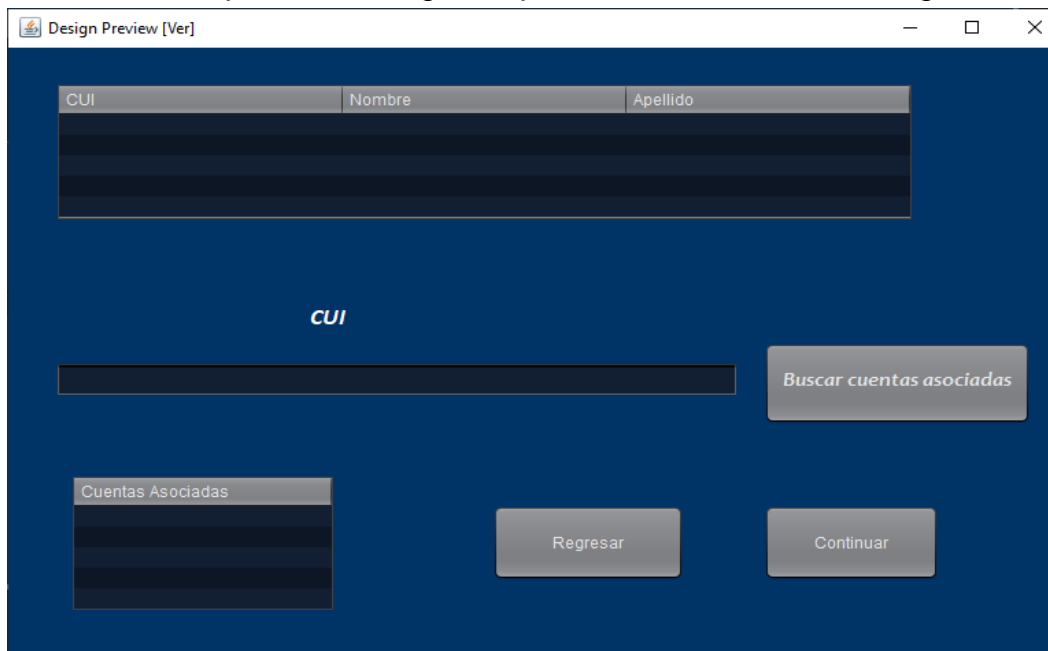
Concluyendo así con los otros dos Buttons llamados “Siguiente” y “regresar” uno servirá para dirigirnos al JFrame de operaciones y el otro como su nombre lo indica regresará al JFrame de cuentas por si el usuario desea generar mas cuentas a su usuario. Para lograr esto fue necesario la implementar setVisible colocando el nombre de nuestra clase principal “Ejemploclase17deagosto” que es donde se alojan las variables de cada JFrame que se utilizó.

```

212 }
213
214 private void SiguienteActionPerformed(java.awt.event.ActionEvent evt) {
215     Ejemploclase17deagosto.hola.setVisible(true);
216     this.setVisible(false);
217 }
218
219 private void regresarActionPerformed(java.awt.event.ActionEvent evt) {
220     Ejemploclase17deagosto.cuen.setVisible(true);
221     this.setVisible(false);
222 }
223
224 /**
225  * @param args the command line arguments
226  */
227 public static void main(String args[]) {
228     /* Set the Nimbus look and feel */
229     Look and feel setting code (optional)
230
231     /* Create and display the form */
232     java.awt.EventQueue.invokeLater(new Runnable() {
233         public void run() {
234             new Ver().setVisible(true);
235         }
236     });
237 }
238
239 }
240
241

```

El diseño de la pantalla emergente que vera el usuario será el siguiente:



### Operaciones:

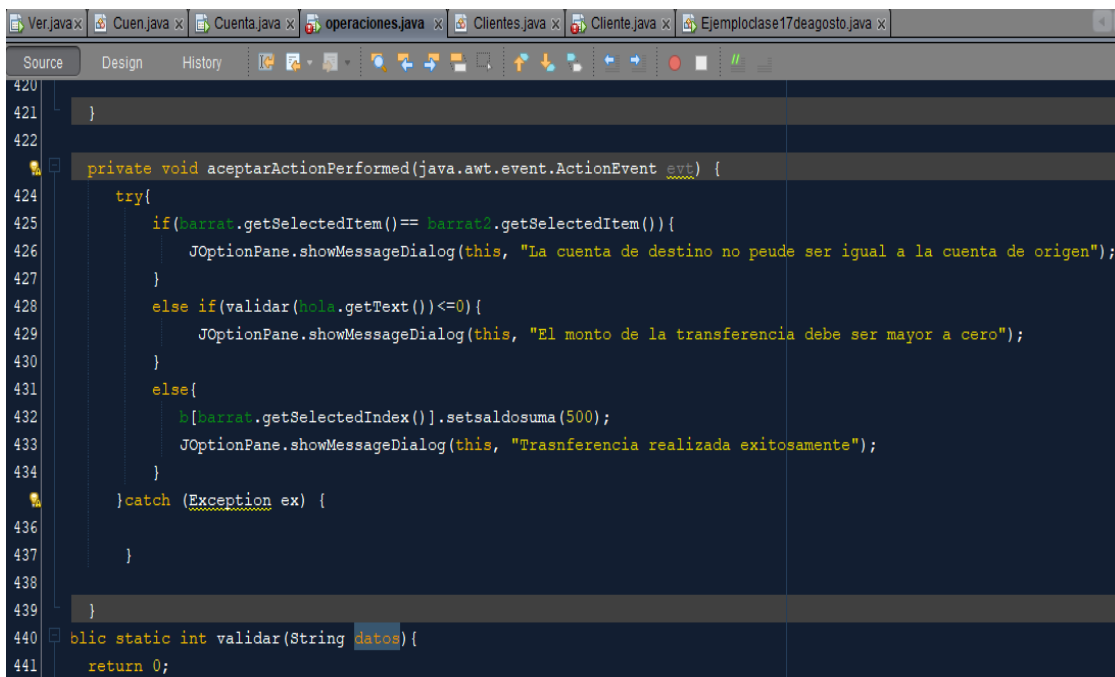
Como siempre se comienza el código cargando la librería `javax.swing.JOptionPane` que nos ayudara para mandar ventanas emergentes a los usuarios.

Declaramos las variables que usaran en esta ocasión como lo son `wey` para Clientes y `b` para Cuentas denominada `Cuen`. También se implemento el uso del `setTitle` y `setLocation` para colocar nombre a la ventana emergente y el otro para su ubicación en la pantalla del usuario.

```
Ver.java x  Cuen.java x  Cuenta.java x  operaciones.java x  Clientes.java x  Cliente.java x  Ejemplod
Source  Design  History
1
2  package ejemploclase17deagosto;
3
4  import javax.swing.JOptionPane;
5
6
7  public class operaciones extends javax.swing.JFrame {
8      Clientes[] wey= new Clientes[5];
9      Cuen[] b= new Cuen[25];
10     int sumar=1;
11     public operaciones() {
12         initComponents();
13         this.setTitle("Operaciones");
14         this.setLocation(400, 220);
15     }
16
17
```

Se implemento el uso del Button denominado “aceptar” que nos ayudara a realizar las operaciones en el JFrame de transacciones donde hicimos uso del if para que, si el usuario ingresaba la misma cuenta le mostrara un mensaje con el JOptionPane donde le decía que la cuenta de destino no puede ser igual a la cuenta de origen, para esto dentro de if de uso el nombre de los ComboBox denominados “barrat” y “barrat2” con el uso del `get.SelectedItem()`.

En el mismo if se hizo uso del else if para validar que el numero ingresado del monto del dinero sea mayor a cero de lo contrario le mostrara un mensaje diciéndole que el montón de la transferencia debe ser mayor a cero, aquí se utilizó `getText` y `JOption`. Y el else se utilizó para descontar el dinero de la cuenta donde salga el dinero, utilizando aquí el `getSelectedIndex` y el `JOptionPane`.



```
420
421
422
423 private void aceptarActionPerformed(java.awt.event.ActionEvent evt) {
424     try{
425         if(barrat.getSelectedItem()== barrat2.getSelectedItem()){
426             JOptionPane.showMessageDialog(this, "La cuenta de destino no puede ser igual a la cuenta de origen");
427         }
428         else if(validar(hola.getText())<=0){
429             JOptionPane.showMessageDialog(this, "El monto de la transferencia debe ser mayor a cero");
430         }
431         else{
432             b[barrat.getSelectedIndex()].setsaldosuma(500);
433             JOptionPane.showMessageDialog(this, "Trasnferencia realizada exitosamente");
434         }
435     }catch (Exception ex) {
436     }
437 }
438
439
440 public static int validar(String datog){
441     return 0;
442 }
```

Se implemento el uso de otro Button llamado dd donde al presionarlo el usuario rellenara de información todos los ComboBox que estén dentro del JFrame operaciones, esto fue posible con el uso de un for y la variable de cuentas “b” en la posición “i”.

De esa manera se logro llamar a cada ComboBox como lo son el de transferencia los cuales son “barrat” y “barrat2”, de pagos llamado “jComboBox5” y el de depósitos llamado “jComboBox1”. También se usó el método `get` para llamar a cada cliente comenzando por su ID seguidamente de su nombre y apellido

```
Ver.java x Cuen.java x Cuenta.java x operaciones.java x Clientes.java x Cliente.java x Ejemploclase17deagosto.java x
Source Design History
454 }
455
456 private void ddActionPerformed(java.awt.event.ActionEvent evt) {
457 try{
458     for(int i=0; i<b.length; i++){
459         b[i]=Ejemploclase17deagosto.cuen.f[i];
460         barrat.addItem(b[i].getID()+" "+b[i].getcliente().getnombre()+" "+b[i].getcliente().getapellido());
461         barrat2.addItem(b[i].getID()+" "+b[i].getcliente().getnombre()+" "+b[i].getcliente().getapellido());
462         jComboBox5.addItem(b[i].getID()+" "+b[i].getcliente().getnombre()+" "+b[i].getcliente().getapellido());
463         jComboBox1.addItem(b[i].getID()+" "+b[i].getcliente().getnombre()+" "+b[i].getcliente().getapellido());
464
465
466     catch (Exception ex) {
467     }
468
469     y{
470         for(int i=0; i<wey.length; i++){
471             wey[i]=Ejemploclase17deagosto.cuen.fabiola[i];
472         }
473     catch (Exception ex) {
474     }
475 }
```

Este código de utilizo para el Button de “Pago de Servicios” llamado “boton2”, usando el método if y else dentro de un getText() que nos ayudara a saber si el numero que ingresan en el JTextfile es mayor que cero, si no es así se les mostrara un mensaje con la función JOptionPane.

```
Ver.java x Cuen.java x Cuenta.java x operaciones.java x Clientes.java x Cliente.java x Ejemploclase17deagosto.java x
Source Design History
478 private void boton2ActionPerformed(java.awt.event.ActionEvent evt) {
479 try{
480     if(validar(text3.getText())<=0){
481         JOptionPane.showMessageDialog(this, "El monto de la transferencia debe ser mayor a cero");
482     }
483     else{
484         JOptionPane.showMessageDialog(this, "Trasnferencia realizada exitosamente");
485     }
486 }catch (Exception ex) {
487 }
488
489 }
```

De la misma manera se utilizó para el Button de “Deposito” llamado “boton1”, usando el método if y else dentro de un getText() que nos ayudara a saber si el número que ingresan en el JTextfile es mayor que cero, si no es así se les mostrara un mensaje con la función JOptionPane.

```
Ver.java x Cuen.java x Cuenta.java x operaciones.java x Clientes.java x Cliente.java x Ejemploclase17deagosto.java x
Source Design History
491 private void boton1ActionPerformed(java.awt.event.ActionEvent evt) {
493     try{
494         if(validar(text1.getText())<=0){
495             JOptionPane.showMessageDialog(this, "El monto de la transferencia debe ser mayor a cero");
496         }
497         else{
498             JOptionPane.showMessageDialog(this, "Trasnferencia realizada exitosamente");
499         }
501     }catch (Exception ex) {
502     }
```

Para concluir con una interfaz de la siguiente manera:

Para transferencias:

Design Preview [operaciones]

Transferencias Pago de Servicios Depositos Historial de Transferencias

*Cuenta de origen*

*Cuenta de destino*

*Monto*

Datos de Cuentas Aceptar

Para Pagos de Servicios:

The screenshot shows a web application window titled "Design Preview [operaciones]". It has a tabbed interface with four tabs: "Transferencias", "Pago de Servicios", "Depositos", and "Historial de Transferencias". The "Pago de Servicios" tab is active. The form has a dark blue background and contains the following elements:

- A dropdown menu labeled **Cuenta a debitar**.
- A dropdown menu labeled **Tipo de servicio** with a list of options: "Luz electrica", "Agua", "Servicio telefonico", and "Internet". The "Luz electrica" option is currently selected and highlighted.
- A text input field labeled **Monto**.
- An "Aceptar" button at the bottom.

Para Depósitos:

The screenshot shows the same web application window, but with the "Depositos" tab selected. The form has a dark blue background and contains the following elements:

- A dropdown menu labeled **Cuenta**.
- A text input field labeled **Monto**.
- An "Aceptar" button at the bottom.



Para el Historial de Transferencia:

Design Preview [operaciones]

Transferencias

Pago de Servicios

Depositos

Historial de Transferencias

Numero unico de cuenta

Mostrar Transacciones

CUI

Nombre

Apellido

ID	Fecha	Detalle	Debito	Credito	Saldo Disponible