

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS

Ing. William Estuardo Escobar Argueta
Aux. Javier Oswaldo Mirón Cifuentes
Aux. Hector Josue Orozco Salazar



USAC.
TRICENTENARIA
Universidad de San Carlos de Guatemala

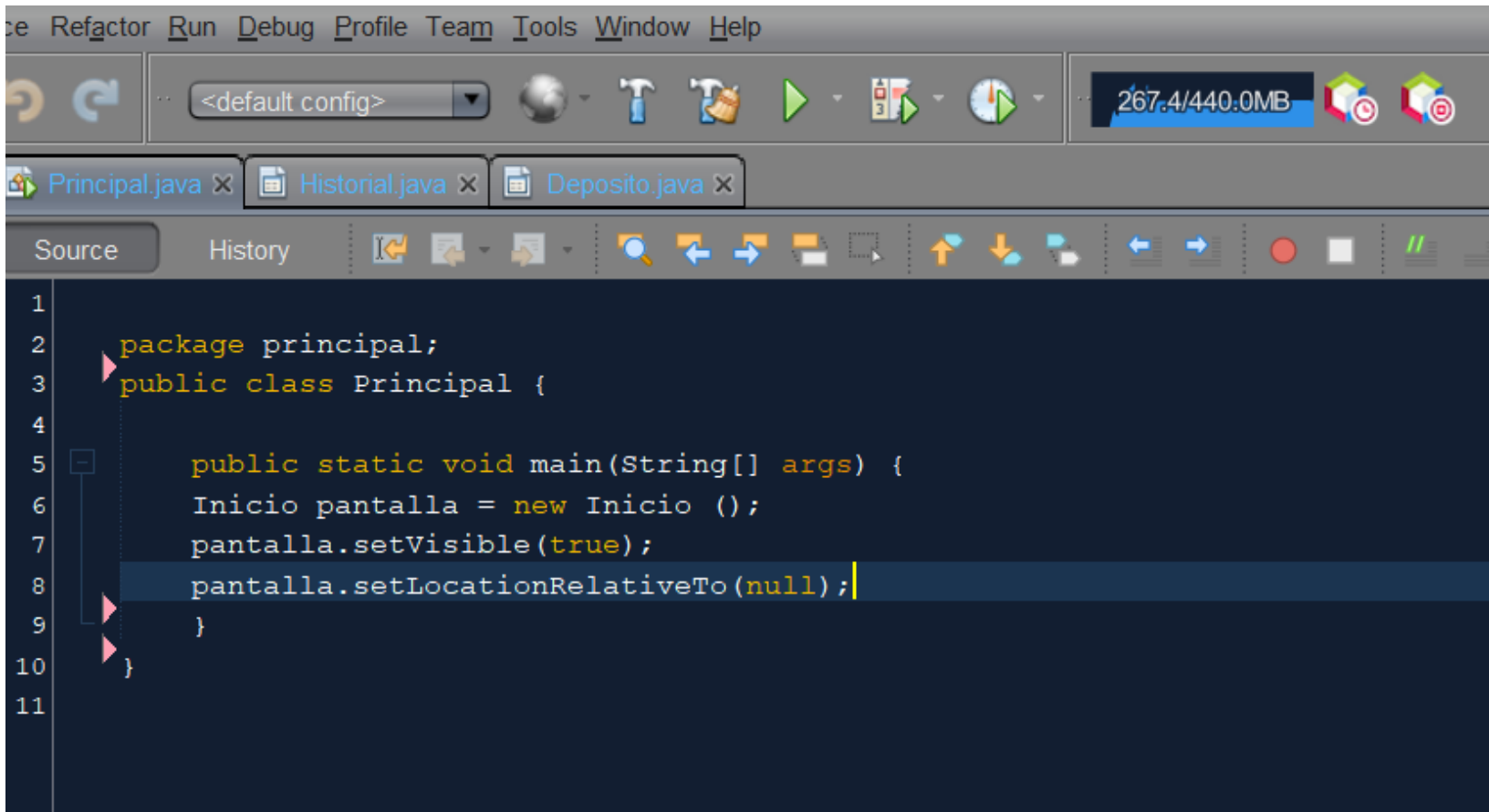


Segundo semestre 2022

Proyecto 1

Clase principal

En la clase principal se encuentra el main, encargado de llamar a la ventana Inicio y junto con esto centrar la posición de la ventana que será mostrada.



```
1
2 package principal;
3 public class Principal {
4
5     public static void main(String[] args) {
6         Inicio pantalla = new Inicio ();
7         pantalla.setVisible(true);
8         pantalla.setLocationRelativeTo(null);
9     }
10 }
11
```

Inicio

En el código inicio solo se encuentra un joptionpanel que es el encargado de mostrar la información del creador y un set visible para llamar a la ventana login.

```
135 private void LoginActionPerformed(java.awt.event.ActionEvent evt) {  
137     Login a = new Login ();  
138     a.setVisible(true);  
139     this.setVisible(false);  
140  
141 }  
142  
143 private void AboutMouseClicked(java.awt.event.MouseEvent evt) {  
145     int p=201901472;  
146     String u="Eliot Oreld Ardón Pérez";  
147     JOptionPane.showMessageDialog(this, " Nombre: " + u + " Carne: " + p );  
148  
149  
150
```

Login

En el código de login podemos encontrar ifs para poder valorar que el usuario y contraseña coincidan con los guardados en las variables, junto con lagunas condicionantes para al momento de dar clic en el text field este pueda borrar su contenido para introducir el nombre de usuario y contraseña correspondientes.

```
187 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
188     // TODO add your handling code here:  
189  
190     Inicio a = new Inicio();  
191     a.setVisible(true);  
192     this.setVisible(false);  
193 }  
194  
195 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
196     // TODO add your handling code here:  
197  
198 }  
199  
200 private void userActionPerformed(java.awt.event.ActionEvent evt) {  
201  
202     if(user.getText().equals("Ingrese su nombre de usuario")){  
203         user.setText("");  
204     }  
205  
206 }  
207  
208 }  
209  
210 private void userMousePressed(java.awt.event.MouseEvent evt) {  
211     // TODO add your handling code here:  
212  
213     if(user.getText().equals("Ingrese su nombre de usuario")){  
214         user.setText("");  
215         user.setForeground(Color.black);  
216     }  
217  
218     if(String.valueOf(paswo.getPassword()).isEmpty()){  
219         paswo.setText("*****");  
220         paswo.setForeground(Color.gray);  
221     }  
222 }
```

```
237 private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {  
238     // TODO add your handling code here:  
239  
240  
241     String usuario="administrador";  
242     int contraseña =201901472;  
243     String con=String.valueOf(contraseña);  
244  
245     String pass =new String (paswo.getPassword());  
246  
247     if(user.getText().equals(usuario) && pass.equals(con)){  
248  
249         Administrador a= new Administrador();  
250         a.setVisible(true);  
251         this.setVisible(false);  
252     }else{  
253  
254         JOptionPane.showMessageDialog(this, "Usuario o Contraseña Incorrecta");  
255     }  
256 }  
257  
258 }  
259  
260 }
```

Administrador

En el administrador se encuentran dos variables enteras que sirven como contadores para la creación de las cuentas, un vector de tipo cliente, junto a sus sets y gets que van a servir para mandar la información a el constructor, a la ventana siempre le asignaremos el `getLocationRelativeTo null` para que aparezcan centradas y un `removeAll` con un `repaint` para ir mostrando los demás paneles según se solicite, junto a esto a cada panel se le pasa la información de los clientes y cuentas asociadas, a través de ventana madre.

```
1
2 package principal;
3
4
5 public class Administrador extends javax.swing.JFrame {
6     public int contador=251;
7     public int id=121;
8     public Cliente clientes []= new Cliente[5];
9
10    public Cliente[] getClientes() {
11        return clientes;
12    }
13
14    public void setClientes(Cliente[] clientes) {
15        this.clientes = clientes;
16    }
17
18    public Administrador() {
19        initComponents();
20        this.setLocationRelativeTo(null);
21        PanelPrincipal pl= new PanelPrincipal();
22        pl.setSize(880, 620);
23        pl.setLocation(0,0);
24
25        content.removeAll();
26        content.add(pl);
27        content.revalidate();
28        content.repaint();
29    }
```

```
private void TransferenciaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Administrador ventanaMadre=this;
    Transferencia pl5= new Transferencia(ventanaMadre);
    pl5.setSize(880, 620);
    pl5.setLocation(0,0);

    content.removeAll();
    content.add(pl5);
    content.revalidate();
    content.repaint();
}
```

Registrar clientes

En el código de registrar clientes pasamos toda la información de los clientes que vayamos a crear con ventana madre la cual pasa toda la información a los demás paneles, por medio de this. Ventana madre y get clientes, para el registro de los clientes se utilizaron if para validar que los campos no estén vacíos, que no hayan y no hayan cuis repetidos y no se exceda de la creación de 5 clientes, junto con esto fors para ir recorriendo el vector donde se guardaran los clientes.

```
1
2 package principal;
3 import javax.swing.JOptionPane;
4
5 public class RegistrarClientes extends javax.swing.JPanel {
6
7     public Cliente clientes [] = new Cliente [5];
8     public Administrador ventanaMadre;
9     public RegistrarClientes(Administrador ventanaMadre) {
10         this.ventanaMadre=ventanaMadre;
11         this.clientes=ventanaMadre.getClientes();
12         initComponents();
13
14     }
15
16 }
```

```
private void CrearActionPerformed(java.awt.event.ActionEvent evt) {
    boolean CuiRepetido= false;
    int contador=0;
    for (int i = 0; i < clientes.length; i++) {
        if (clientes[i]!=null) {
            contador =contador+1;
        }
    }
    if (contador==clientes.length) {
        JOptionPane.showMessageDialog(this,"limite de clientes alcanzado");
    }else{
        if (!txtNombre.getText().isEmpty() && !txtCui.getText().isEmpty() && !txtApellido.getText().isEmpty()) {
            String ClienteCui = txtCui.getText();
            String ClienteNombre = txtNombre.getText();
            String ClienteApellido = txtApellido.getText();
            for (int j = 0; j < clientes.length; j++) {
                if (clientes[j]!=null) {
                    if (clientes[j].getDocumentoDeIdentificacion()==Long.parseLong(ClienteCui)) {
                        CuiRepetido=true;
                        JOptionPane.showMessageDialog(this,"Cui repetido");
                        break;
                    }
                }
            }
        }
    }
}
```

Crear cuenta asociada

En el código de crear una cuenta asociada tenemos 5 ifs para validar que a cada cliente tenga cuentas asociadas y si así lo es mostrar esas cuentas asociadas en un combo box, para crear las asociadas se valida la posición del cliente en el vector con un if y después se recorre otro vector que está asociado a el de clientes para guardar las cuentas asociadas de el mismo, hay 2 variables enteras para asignarle un id y un saldo inicial de cero, y otra condicionante para valuar que solo se creen 5 asociadas a cada cliente.

```
public class CrearUnaCuenta extends javax.swing.JPanel {
    Cliente clientes [] = new Cliente (5);
    public Administrador Madre;

    public CrearUnaCuenta(Administrador ventanaMadre) {
        this.clientes=ventanaMadre.getClientes();
        this.Madre=ventanaMadre;
        initComponents();

        if(clientes[0]!=null){
            cboClientes.removeAllItems();
            cboClientes.addItem(clientes[0].getDocumentoDeIdentificacion() + " - " + clientes[0].getNombre() + " " + clientes[0].getApellido());
        }
        if(clientes[1]!=null){
            cboClientes.removeAllItems();
            cboClientes.addItem(clientes[0].getDocumentoDeIdentificacion() + " - " + clientes[0].getNombre() + " " + clientes[0].getApellido());
            cboClientes.addItem(clientes[1].getDocumentoDeIdentificacion() + " - " + clientes[1].getNombre() + " " + clientes[1].getApellido());
        }
        if(clientes[2]!=null){
            cboClientes.removeAllItems();
            cboClientes.addItem(clientes[0].getDocumentoDeIdentificacion() + " - " + clientes[0].getNombre() + " " + clientes[0].getApellido());
            cboClientes.addItem(clientes[1].getDocumentoDeIdentificacion() + " - " + clientes[1].getNombre() + " " + clientes[1].getApellido());
            cboClientes.addItem(clientes[2].getDocumentoDeIdentificacion() + " - " + clientes[2].getNombre() + " " + clientes[2].getApellido());
        }
        if(clientes[3]!=null){
            cboClientes.removeAllItems();
            cboClientes.addItem(clientes[0].getDocumentoDeIdentificacion() + " - " + clientes[0].getNombre() + " " + clientes[0].getApellido());
            cboClientes.addItem(clientes[1].getDocumentoDeIdentificacion() + " - " + clientes[1].getNombre() + " " + clientes[1].getApellido());
            cboClientes.addItem(clientes[2].getDocumentoDeIdentificacion() + " - " + clientes[2].getNombre() + " " + clientes[2].getApellido());
            cboClientes.addItem(clientes[3].getDocumentoDeIdentificacion() + " - " + clientes[3].getNombre() + " " + clientes[3].getApellido());
        }
        if(clientes[4]!=null){
            cboClientes.removeAllItems();
            cboClientes.addItem(clientes[0].getDocumentoDeIdentificacion() + " - " + clientes[0].getNombre() + " " + clientes[0].getApellido());
            cboClientes.addItem(clientes[1].getDocumentoDeIdentificacion() + " - " + clientes[1].getNombre() + " " + clientes[1].getApellido());
            cboClientes.addItem(clientes[2].getDocumentoDeIdentificacion() + " - " + clientes[2].getNombre() + " " + clientes[2].getApellido());
            cboClientes.addItem(clientes[3].getDocumentoDeIdentificacion() + " - " + clientes[3].getNombre() + " " + clientes[3].getApellido());
            cboClientes.addItem(clientes[4].getDocumentoDeIdentificacion() + " - " + clientes[4].getNombre() + " " + clientes[4].getApellido());
        }
    }
}
```

```
private void btnCrearActionPerformed(java.awt.event.ActionEvent evt) {
    int saldo=0;
    int contador=0;
    int posicionClienteSeleccionado=cboClientes.getSelectedIndex();
    if (clientes[posicionClienteSeleccionado]!=null) {
        for (int i = 0; i < clientes[posicionClienteSeleccionado].listadoDeCuentasAsociadas.length; i++) {
            if (clientes[posicionClienteSeleccionado].listadoDeCuentasAsociadas[i].getDisponibilidad()==false) {
                contador =contador + 1;
            }
        }
        if (contador == clientes.length) {
            JOptionPane.showMessageDialog(this,"limite de cuentas asociadas alcanzado");
        }else{
            if (clientes[posicionClienteSeleccionado] != null) {
                for (int j = 0; j < clientes[posicionClienteSeleccionado].listadoDeCuentasAsociadas.length; j++) {
                    if (clientes[posicionClienteSeleccionado].listadoDeCuentasAsociadas[j].Disponibilidad==true) {
                        int ContadorDeElCliente=Madre.contador;
                        Madre.contador+=1;
                        clientes[posicionClienteSeleccionado].listadoDeCuentasAsociadas[j]=new CuentaAsociada(ContadorDeElCliente, saldo);
                        break;
                    }
                }
            }
        }
    }
}
```


Visualizar informes de clientes

En el código de visualizar informes de clientes podemos encontrar varios ifs para poder validar que haya clientes creados y de así ser irlos mostrando, luego contamos con otro if para validar que el cui ingresado este asociado a una cuenta y si así los es utilizar fors para ir mostrando las cuentas asociadas a ese cui.

```
TbClientes.setBorder(null);
if(clientes[0]!=null){
TbClientes.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {clientes[0].getDocumentoDeIdentificacion(),clientes[0].getNombre(),clientes[0].getApellido()},
        {null,null,null},
        {null,null,null},
        {null,null,null},
        {null,null,null}
    },
    new String [] {
        "CUI", "Nombre", "Apellido"
    }
));
if(clientes[1]!=null){
TbClientes.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {clientes[0].getDocumentoDeIdentificacion(),clientes[0].getNombre(),clientes[0].getApellido()},
        {clientes[1].getDocumentoDeIdentificacion(),clientes[1].getNombre(),clientes[1].getApellido()},
        {null,null,null},
        {null,null,null},
        {null,null,null}
    },
    new String [] {
        "CUI", "Nombre", "Apellido"
    }
));
}
if(clientes[2]!=null){
TbClientes.setModel(new javax.swing.table.DefaultTableModel(
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
249     String Busqueda=tctBusqueda.getText();
250     model=new DefaultTableModel();
251     model.addColumn("Cuentas Asociadas");
252
253     for (int i = 0; i < clientes.length; i++) {
254         if (clientes[i]!=null) {
255             if (clientes[i].getDocumentoDeIdentificacion()==Long.parseLong(Busqueda)) {
256                 for (int j = 0; j < clientes[i].listadoDeCuentasAsociadas.length; j++) {
257                     if (clientes[i].listadoDeCuentasAsociadas[j].Disponibilidad==false) {
258                         model.addRow(new Object[]{clientes[i].listadoDeCuentasAsociadas[j].NumeroDeCuentaAsociada});
259                     }
260                 }
261                 jtAsociados.setModel(model);
262                 break;
263             }
264         }
265     }
266 }
267
268 // Variables declaration - do not modify
```


Deposito

En el código de depósito tenemos un if con dos for los cuales servirán para ir recorriendo el cliente y las cuentas asociadas en caso de que encuentre serán mostradas en el combo box, para hacer el depósito se validara con un if que el monto no sea de 0 ni el campo este vacío, luego se valuará que el id del asociado coincida con uno creado para luego con dos for buscarlo y acreditarle el monto ingresado, también se utilizaron algunos parseos para transformar variables.

```
1
2 package principal;
3 import java.util.Calendar;
4 import java.util.GregorianCalendar;
5 import javax.swing.JOptionPane;
6
7 public class Deposito extends javax.swing.JPanel {
8     Cliente clientes [] = new Cliente [5];
9     public Administrador Madre;
10    public Deposito(Administrador ventanaMadre) {
11        this.clientes=ventanaMadre.getClientes();
12        this.Madre=ventanaMadre;
13        initComponents();
14
15        for (int i = 0; i < clientes.length; i++) {
16            for (int j = 0; j < clientes[i].listadoDeCuentasAsociadas.length; j++) {
17                if (clientes[i] != null) {
18                    if (clientes[i].listadoDeCuentasAsociadas[j].Disponibilidad == false) {
19                        jcbAsociados.addItem(Integer.toString(clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada()) + " - " + " Cuenta de");
20                    }
21                }
22            }
23        }
24    }
25 }
```

```
private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    if (jtxtMonto.getText().isEmpty() || Integer.parseInt(jtxtMonto.getText()) == 0) {
        JOptionPane.showMessageDialog(this, "El monto debe ser superior a 0");
    } else {
        String dato = String.valueOf(jcbAsociados.getSelectedIndex());
        String[] fragmento = dato.split("");
        int NoAsociado = Integer.parseInt(fragmento[0]);
        int Monto = Integer.parseInt(jtxtMonto.getText());
        for (int i = 0; i < clientes.length; i++) {
            if (clientes[i] != null) {
                for (int j = 0; j < clientes[i].listadoDeCuentasAsociadas.length; j++) {
                    if (clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada() == NoAsociado) {
                        int ahora = clientes[i].listadoDeCuentasAsociadas[j].getSalado();
                        clientes[i].listadoDeCuentasAsociadas[j].setSalado(ahora + Monto);
                        JOptionPane.showMessageDialog(this, "Deposito realizado exitosamente");
                        System.out.println(clientes[i].listadoDeCuentasAsociadas[j].getSalado());
                        jtxtMonto.setText("");
                    }
                }
            }
        }
    }
}
```

Transferencia

En el código de transferencia tenemos un if con dos for los cuales servirán para ir recorriendo el cliente y las cuentas asociadas en caso de que encuentre serán mostradas en los combo box, para hacer la transferencia de una cuenta a otra primero se validara con un if que el monto sea superior a 0 y la cantidad que quiere transferir mayor o igual a la cantidad que tiene la cuenta, después con otro if se validara que la cuenta de destino sea diferente a la de origen cuanto esto se cumpla se utilizaran fors para recorrer ambas cuentas y transferir de una a otra los fondos solicitados.

```
5
6 public class Transferencia extends javax.swing.JPanel {
7     Cliente clientes [] = new Cliente [5];
8     public Administrador Madre;
9     public Transferencia(Administrador ventanaMadre) {
10         this.clientes=ventanaMadre.getClientes();
11         this.Madre=ventanaMadre;
12         initComponents();
13
14         for (int i = 0; i < clientes.length; i++) {
15             for (int j = 0; j < clientes.length; j++) {
16                 if (clientes[i]!=null) {
17                     if (clientes[i].listadoDeCuentasAsociadas[j].Disponibilidad==false) {
18                         jcbOrigen.addItem(Integer.toString(clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada()) + " - " + " Cuenta de: " + clientes[i].listadoDeCuentasAsociadas[j].getSalado());
19                         jcbDestino.addItem(Integer.toString(clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada()) + " - " + " Cuenta de: " + clientes[i].listadoDeCuentasAsociadas[j].getSalado());
20                     }
21                 }
22             }
23         }
24     }
25 }
26
27
```

```
153 if (jtxtMonto.getText().isEmpty() || Integer.parseInt(jtxtMonto.getText())==0) {
154     JOptionPane.showMessageDialog(this,"El monto debe ser superior a 0");
155 }else{
156     String dato=String.valueOf(jcbOrigen.getSelectedItem());
157     String[] fragmento= dato.split("");
158     int NoAsociado=Integer.parseInt(fragmento[0]);
159     int Monto=Integer.parseInt(jtxtMonto.getText());
160
161     String dato1=String.valueOf(jcbDestino.getSelectedItem());
162     String[] fragmento1=dato1.split("");
163     int NoAsociado1=Integer.parseInt(fragmento1[0]);
164
165     if (NoAsociado!=NoAsociado1) {
166         for (int k = 0; k < clientes.length; k++) {
167             for (int i = 0; i < clientes.length; i++) {
168                 if (clientes[k]!=null) {
169                     if (clientes[i]!=null) {
170                         for (int l = 0; l < clientes[k].listadoDeCuentasAsociadas.length; l++) {
171                             for (int j = 0; j < clientes[i].listadoDeCuentasAsociadas.length; j++) {
172
173                                 if (clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada()==NoAsociado && clientes[k].listadoDeCuentasAsociadas[l].getNumeroDeCuentaAsociada()==NoAsociado1) {
174                                     if (clientes[i].listadoDeCuentasAsociadas[j].getSalado()>=Monto) {
175
176                                         int ahora=clientes[i].listadoDeCuentasAsociadas[j].getSalado();
177                                         int ahora1=clientes[k].listadoDeCuentasAsociadas[l].getSalado();
178                                         clientes[i].listadoDeCuentasAsociadas[j].setSalado(ahora - Monto);
179                                         clientes[k].listadoDeCuentasAsociadas[l].setSalado(ahora1 + Monto);
180
181                                         JOptionPane.showMessageDialog(this,"Transferencia realizada exitosamente");
182                                         System.out.println(clientes[i].listadoDeCuentasAsociadas[j].getSalado());
183                                         System.out.println(clientes[k].listadoDeCuentasAsociadas[l].getSalado());
184                                         jtxtMonto.setText("");
185                                         break;
186                                     }
187                                 }
188                             }
189                         }
190                     }
191                 }
192             }
193         }
194     }
195 }
```

Pago de servicios

el código de pago de servicios primero se utiliza dos for para recorrer el vector de clientes y el vector de asociados, para mostrarlos en el combo box se valúa que el cliente exista y tenga asociados, si se cumple se muestra en el combo box. Para poder realizar el pago del servicio seleccionado, primero se valúa con un if que el monto sea superior a 0, si se cumple entra a un switch donde se buscara la condición correspondiente a el servicio seleccionado en el cual se utilizaron dos for para buscar el número de cliente y desde el mismo hacer el depósito a el servicio correspondiente, también habrá otro if para evaluar si tiene el saldo necesario para realizar el pago, si se cumple automáticamente se realiza el pago.

```
1 package principal;
2
3
4 import javax.swing.JOptionPane;
5
6 public class Pagos extends javax.swing.JPanel {
7     Cliente clientes [] = new Cliente [5];
8     public Administrador Madre;
9     public Pagos(Administrador ventanaMadre) {
10         this.clientes=ventanaMadre.getClientes();
11         this.Madre=ventanaMadre;
12         initComponents();
13
14         for (int i = 0; i < clientes.length; i++) {
15             for (int j = 0; j < clientes[i].listadoDeCuentasAsociadas.length; j++) {
16                 if (clientes[i] != null) {
17                     if (clientes[i].listadoDeCuentasAsociadas[j].Disponibilidad == false) {
18                         jcbOrigen.addItem(Integer.toString(clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada()) + " - " + " Cuenta de
19                     }
20                 }
21             }
22         }
23     }
24 }
25
26 @SuppressWarnings("unchecked")
27 Generated Code
28
29
30 private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
31
32     if (jtxtMonto.getText().isEmpty() || Integer.parseInt(jtxtMonto.getText()) == 0) {
33         JOptionPane.showMessageDialog(this, "El monto debe ser superior a 0");
34     }
35 }
```

```
143
144 if (jtxtMonto.getText().isEmpty() || Integer.parseInt(jtxtMonto.getText()) == 0) {
145     JOptionPane.showMessageDialog(this, "El monto debe ser superior a 0");
146 } else {
147     String dato = String.valueOf(jcbOrigen.getSelectedItem());
148     String[] fragmento = dato.split("");
149     int NoAsociado = Integer.parseInt(fragmento[0]);
150     int Monto = Integer.parseInt(jtxtMonto.getText());
151
152     String destino = String.valueOf(jcbServicio.getSelectedItem());
153     System.out.println(destino);
154
155     switch (destino) {
156         case "Luz electrica":
157             for (int i = 0; i < clientes.length; i++) {
158                 if (clientes[i] != null) {
159                     for (int j = 0; j < clientes[i].listadoDeCuentasAsociadas.length; j++) {
160                         if (clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada() == NoAsociado) {
161                             if (clientes[i].listadoDeCuentasAsociadas[j].getSaldo() >= Monto) {
162
163                                 int ahora = clientes[i].listadoDeCuentasAsociadas[j].getSaldo();
164                                 clientes[i].listadoDeCuentasAsociadas[j].setSaldo(ahora - Monto);
165                                 int Elec = clientes[i].listadoDeCuentasAsociadas[j].servicio.getLuzElectrica();
166                                 clientes[i].listadoDeCuentasAsociadas[j].servicio.setLuzElectrica(Elec + Monto);
167                                 JOptionPane.showMessageDialog(this, "Deposito realizado exitosamente");
168                                 System.out.println(clientes[i].listadoDeCuentasAsociadas[j].getSaldo());
169                                 System.out.println(clientes[i].listadoDeCuentasAsociadas[j].servicio.getLuzElectrica());
170                                 jtxtMonto.setText("");
171                                 break;
172                             }
173                         }
174                     }
175                 }
176             }
177         }
178     }
179 }
```

Historial de transacciones

En el código de historial de transacciones tenemos un if que valua que el id ingresado exista y si así lo es entrara en dos for que recorrerán el cliente y las cuentas asociadas en busca del cliente al que le pertenece dicha cuenta u lo mostrara por medio de de los set de clientes y su información.

```
172 private void jbtnMostrarActionPerformed(java.awt.event.ActionEvent evt) {  
174  
175     if (jtxtId.getText().isEmpty() || Integer.parseInt(jtxtId.getText())==0) {  
176         JOptionPane.showMessageDialog(this,"No se encontro ningun cliente asociado con ese id");  
177     }else{  
178         for (int i = 0; i < clientes.length; i++) {  
179             if (clientes[i]!=null) {  
180                 for (int j = 0; j < clientes[i].listadoDeCuentasAsociadas.length; j++) {  
181                     if (clientes[i].listadoDeCuentasAsociadas[j].getNumeroDeCuentaAsociada()==Integer.valueOf(jtxtId.getText())) {  
182                         jtxtCui.setText(String.valueOf(clientes[i].getDocumentoDeIdentificacion()));  
183                         jtxtNombre.setText(clientes[i].getNombre());  
184                         jtxtApellido.setText(clientes[i].getApellido());  
185                     }  
186                 }  
187             }  
188         }  
189     }  
190 }
```

Cientes

Para clientes tenemos un constructor con parámetros y otro vacío los cuales cuentan con variables necesarias para el registro de los clientes, junto con esto cada variable cuenta con sus get y set.

```
1
2 package principal;
3
4 public class Cliente {
5     public long DocumentoDeIdentificacion;
6     public String Nombre;
7     public String Apellido;
8
9
10    //Constructor
11    public CuentaAsociada listadoDeCuentasAsociadas[] = {new CuentaAsociada(), new CuentaAsociada(), new CuentaAsociada(), new CuentaAsociada(), new CuentaAsociada()};
12
13    public Cliente() {}
14
15    public Cliente(long DocumentoDeIdentificacion, String Nombre, String Apellido) {
16        this.DocumentoDeIdentificacion = DocumentoDeIdentificacion;
17        this.Nombre = Nombre;
18        this.Apellido = Apellido;
19    }
20
21    // Asociar un Cuenta
22    public void AsociarCuenta(int saldo) {
23        for (int i = 0; i < this.listadoDeCuentasAsociadas.length; i++) {
24            if (this.listadoDeCuentasAsociadas[i].Disponibilidad == true) {
25                System.out.println("espacio libre");
26                this.listadoDeCuentasAsociadas[i].setSalado(saldo);
27                this.listadoDeCuentasAsociadas[i].setDisponibilidad(false);
28                break;
29            } else {
30                System.out.println("espacio ocupado");
31            }
32        }
33    }
34
35 }
```

```
34
35
36 // Informacion de Cuentas asociadas
37 public void informacionDeCuentasAsociadas() {
38     for (int i = 0; i < this.listadoDeCuentasAsociadas.length; i++) {
39         if (this.listadoDeCuentasAsociadas[i].Disponibilidad == false) {
40             System.out.println("espacio libre");
41             this.listadoDeCuentasAsociadas[i].getNumeroDeCuentaAsociada();
42             this.listadoDeCuentasAsociadas[i].getSalado();
43
44         } else {
45             System.out.println("espacio ocupado");
46         }
47     }
48 }
49
50
51 }
52
53
54
55
56 // Geter and Seter
57
58 public CuentaAsociada[] getListadoDeCuentasAsociadas() {
59     return listadoDeCuentasAsociadas;
60 }
61
62 public void setListadoDeCuentasAsociadas(CuentaAsociada[] listadoDeCuentasAsociadas) {
63     this.listadoDeCuentasAsociadas = listadoDeCuentasAsociadas;
64 }
65
66
67 public long getDocumentoDeIdentificacion() {
68     return DocumentoDeIdentificacion;
69 }
```

Cuenta asociada

En cuenta asociada contamos con dos constructores uno vacío y otro lleno los cuales son necesarios para la creación de cuentas asociadas, tienen los parámetros necesarios para dicha creación junto con sus get y set.

```
1
2 package principal;
3
4 public class CuentaAsociada {
5     int Saldo;
6     int NumeroDeCuentaAsociada;
7     boolean Disponibilidad = true;
8
9     Servicios servicio = new Servicios();
10    Transacciones info [] = new Transacciones[4];
11
12    public boolean getDisponibilidad() {
13        return Disponibilidad;
14    }
15
16    public void setDisponibilidad(boolean Disponibilidad) {
17        this.Disponibilidad = Disponibilidad;
18    }
19
20    CuentaAsociada() {
21        this.Saldo=0;
22        this.NumeroDeCuentaAsociada=0;
23        this.Disponibilidad=true;
24    }
25
26    CuentaAsociada(int NumeroDeCuentaAsociada,int Saldo ){
27        this.Saldo=Saldo;
28        this.NumeroDeCuentaAsociada=NumeroDeCuentaAsociada;
29        this.Disponibilidad=false;
30    }
31
32
33    public int getSaldo() {
34        return Saldo;
35    }
36
```

Servicios

En servicios solo contamos con un constructor vacío el cual dispone de la capacidad de guardar los valores de las variables con las que cuenta, junto con esto cada variable cuenta con sus get y set.

```
1
2 package principal;
3
4 public class Servicios {
5     int LuzElectrica=0;
6     int Agua=0;
7     int ServicioTelefonico=0;
8
9     public Servicios() {
10    }
11
12     public int getLuzElectrica() {
13         return LuzElectrica;
14     }
15
16     public void setLuzElectrica(int LuzElectrica) {
17         this.LuzElectrica = LuzElectrica;
18     }
19
20     public int getAgua() {
21         return Agua;
22     }
23
24     public void setAgua(int Agua) {
25         this.Agua = Agua;
26     }
27
28     public int getServicioTelefonico() {
29         return ServicioTelefonico;
30     }
31
32     public void setServicioTelefonico(int ServicioTelefonico) {
33         this.ServicioTelefonico = ServicioTelefonico;
34     }
35 }
```