S09 – PRA Estructura de Datos

Sesión 10: Estructuras no lineales: Árboles

I. OBJETIVOS

Al término de esta experiencia, el estudiante será capaz de:

1. Emplear Estructuras Dinámicas para organizar y manipular sus datos

II. EQUIPOS Y MATERIALES

- Computador
- Guía de Laboratorio

III. METODOLOGIA Y ACTIVIDADES

a) Teoría de Árboles

IV. IMPORTANTE

Antes de iniciar con el desarrollo del Laboratorio, crearemos siempre, una carpeta, donde se guardará toda la información del presente laboratorio. Para ello realice lo siguiente:

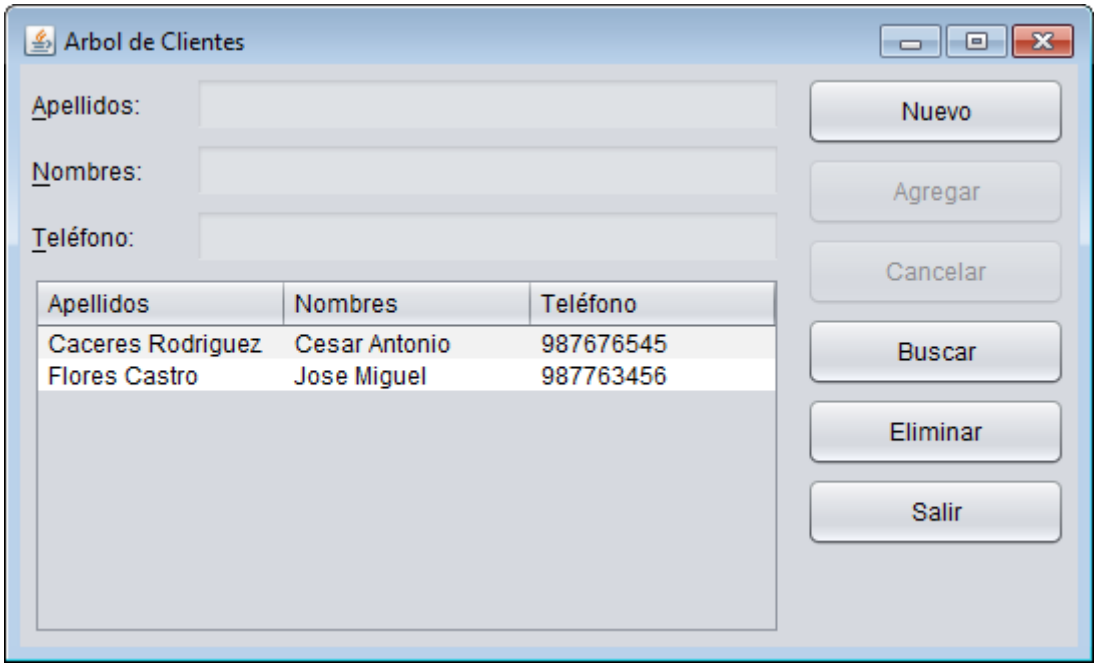
- Ingrese al Explorador del Windows (puede hacerlo dando clic derecho sobre el Botón Inicio de la Barra de Tareas y seleccione la opción Explorar).
- La ventana del Explorador esta dividida en dos columnas, en la columna de la izquierda busque hacia abajo la unidad de almacenamiento (D:) y de un clic izquierdo sobre él. Luego dirija el mouse hacia la columna de la derecha y en un sector vacío, presione clic derecho, seleccione la opción Nuevo y luego la opción Carpeta.
- Aparecerá una carpeta amarilla con un texto: Nueva Carpeta sombreado en azul, digite sobre él, el nombre para su carpeta (este puede ser **S10_ESDA_Apellidos**), luego de digitar presione la tecla Enter. Listo, ya tiene su carpeta dentro de la cual guardará todo lo que trabaje a continuación.
- Cierre la ventana del Explorador del Windows.

V. PROCEDIMIENTO

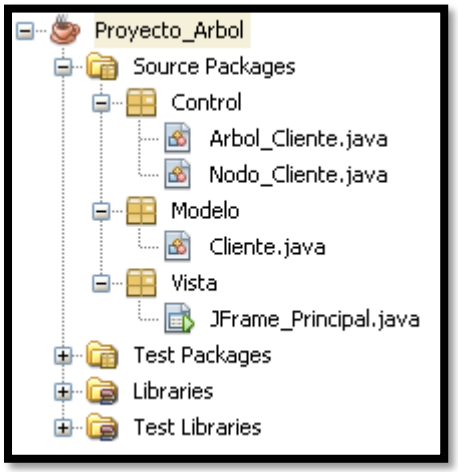
a) Encender el computador.
b) Crear carpeta donde guardará el documento son su información.
c) Ingresar al software NetBeans IDE y allí crear el proyecto solicitado

Ejercicio N° 1:

Elabore un proyecto que permita registrar dentro de una **Árbol** los datos (Apellidos, Nombres y Teléfono) de un grupo de clientes. Asimismo, deberá de permitir realizar diversas operaciones como: Agregar, listar, buscar y eliminar.



1. Cree un proyecto con la siguiente estructura



2. Implemente el código de la clase llamada **Cliente**.

```
1  package Modelo;
2
3  public class Cliente
4  {
5      private String Apellidos;
6      private String Nombres;
7      private String Telefono;
8
```

```
9       public Cliente(Object[] Registo)
10      {
11          this.Apellidos = Registo[0].toString();
12          this.Nombres = Registo[1].toString();
13          this.Telefono = Registo[2].toString();
14      }
15
16      public Object[] getRegistro()
17      {
18          Object[] Registro = {Apellidos, Nombres, Telefono};
19          return Registro;
20      }
21
22      public String getNomCompleto() {return Apellidos + " " + Nombres;};
23
24      public String getApellidos() {return Apellidos;}
25      public void setApellidos(String Apellidos) {this.Apellidos = Apellidos;}
26
27      public String getNombres() {return Nombres;}
28      public void setNombres(String Nombres) {this.Nombres = Nombres;}
29
30      public String getTelefono() {return Telefono;}
31      public void setTelefono(String Telefono) {this.Telefono = Telefono;}
32  }
```

3. Implemente el código de la clase llamada **Nodo_Cliente**.

```
1   package Control;
2
3   import Modelo.Cliente;
4
5   public class Nodo_Cliente
6   {
7       private Cliente Elemento;
8       private Nodo_Cliente Izq, Der;
9
10      public Nodo_Cliente(Cliente Elemento)
11      {
12          this.Elemento = Elemento;
13          Izq = Der = null;
14      }
15
16      public Nodo_Cliente getDer() {return Der;}
17      public void setDer(Nodo_Cliente Der) {this.Der = Der;}
18
19      public Cliente getElemento() {return Elemento;}
20      public void setElemento(Cliente Elemento) {this.Elemento = Elemento;}
21
22      public Nodo_Cliente getIzq() {return Izq;}
23      public void setIzq(Nodo_Cliente Izq) {this.Izq = Izq;}
24
25  }
```

4. Implemente el código de la clase llamada **Arbol_Cliente**.
   a. Implemente los **atributos**, **constructor** y métodos **Getter** y **Setter**

```
1   package Control;
2
3   import Modelo.Cliente;
4   import javax.swing.table.DefaultTableModel;
5
6   public class Arbol_Cliente
7   {
8       private Nodo_Cliente Raiz;
9
10      public Arbol_Cliente()
11      {
12          Raiz = null;
13      }
14
15      public void setRaiz(Nodo_Cliente Raiz) { this.Raiz = Raiz; }
16
17      public Nodo_Cliente getRaiz() { return Raiz; }
18
```

   b. Implemente el método **Agregar**

```
19      public Nodo_Cliente Agregar(Nodo_Cliente Nodo, Cliente Elemento)
20      {
21          if(Nodo == null)
22          {
23              Nodo_Cliente Nuevo = new Nodo_Cliente(Elemento);
24              return Nuevo;
25          }
26          else
27          {
28              if(Elemento.getNomCompleto().compareTo
29                                      (Nodo.getElemento().getNomCompleto())>0)
30              {
31                  Nodo.setDer(Agregar(Nodo.getDer(), Elemento));
32              }
33              else
34              {
35                  Nodo.setIzq(Agregar(Nodo.getIzq(), Elemento));
36              }
37          }
38          return Nodo;
39
40      }
```

c. Implemente el método **BuscarApeNom**

```
42      public Nodo_Cliente BuscarApeNom(String Dato)
43      {
44          Nodo_Cliente Auxiliar = Raiz;
45
46          while(Auxiliar != null)
47          {
48              if(Auxiliar.getElemento().getNomCompleto().startsWith(Dato))
49              {
50                  return Auxiliar;
51              }
52              else
53              {
54                  if(Dato.compareTo(Auxiliar.getElemento().getNomCompleto())>0)
55                      Auxiliar = Auxiliar.getDer();
56                  else
57                      Auxiliar = Auxiliar.getIzq();
58              }
59          }
60          return null;
61      }
```

d. Implemente el método **Listar_InOrder**

```
63      public void Listar_InOrder(Nodo_Cliente Nodo, DefaultTableModel modTabla)
64      {
65          if(Nodo != null)
66          {
67              Listar_InOrder(Nodo.getIzq(), modTabla);
68              modTabla.addRow(Nodo.getElemento().getRegistro());
69              Listar_InOrder(Nodo.getDer(), modTabla);
70          }
71      }
```

e. Implemente el método **BuscarMayorIzquierda**

```
73      public Nodo_Cliente BuscarMayorIzquierda(Nodo_Cliente Auxiliar)
74      {
75          if(Auxiliar != null)
76          {
77              while(Auxiliar.getDer() != null)
78              {
79                  Auxiliar = Auxiliar.getDer();
80              }
81          }
82          return Auxiliar;
83      }
```

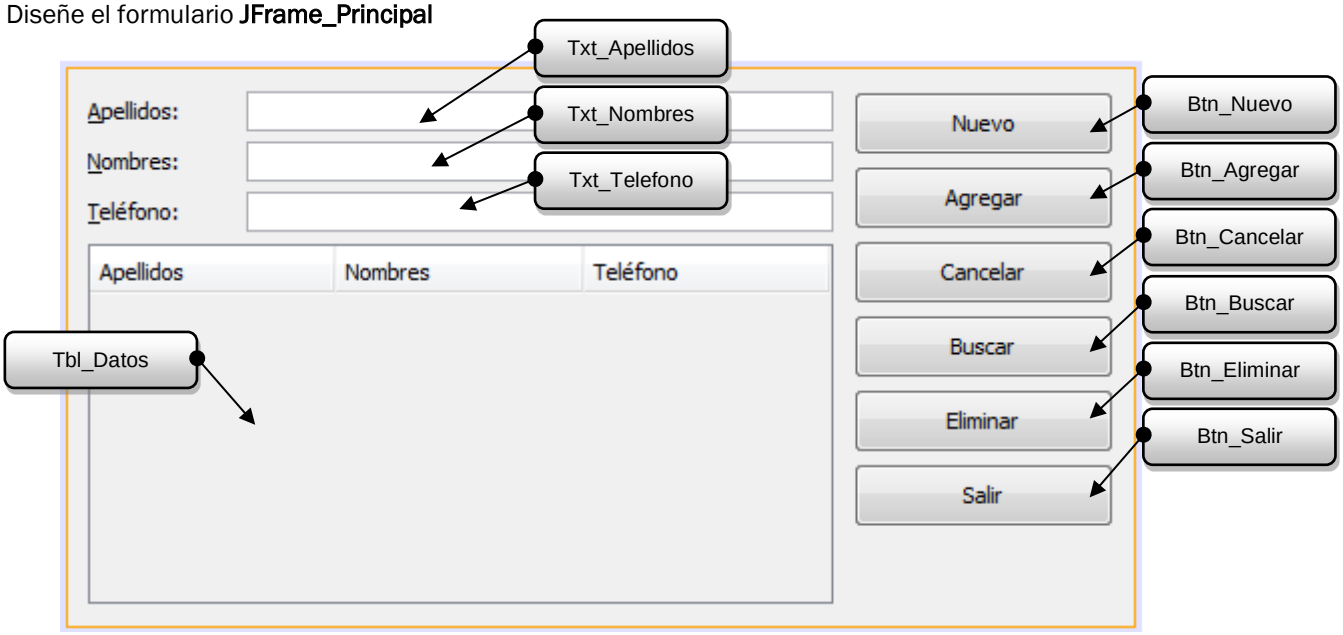
f. Implemente el método **EliminarMayorIzquierda**

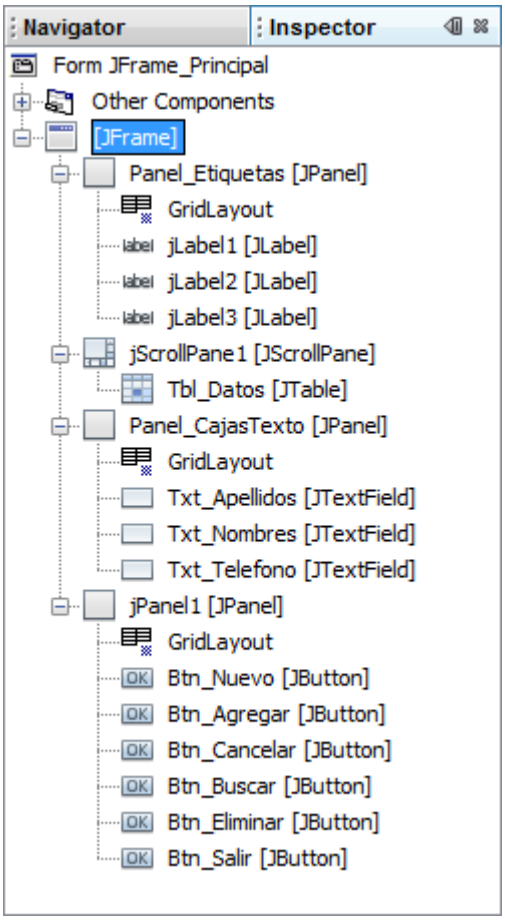
```
85      public Nodo_Cliente EliminarMayorIzquierda(Nodo_Cliente Auxiliar)
86      {
87          if(Auxiliar == null)
88              return null;
89          else if(Auxiliar.getDer() != null)
90          {
91              Auxiliar.setDer(EliminarMayorIzquierda(Auxiliar.getDer()));
92              return Auxiliar;
93          }
94          return Auxiliar.getIzq();
95      }
```

g. Implemente el método **Eliminar**

```
97          public Nodo_Cliente Eliminar(Nodo_Cliente Auxiliar, String Dato)
98          {
99              if(Auxiliar == null)
100                 return null;
101
102             if (Dato.compareTo(Auxiliar.getElemento().getNomCompleto()) < 0)
103             {
104                 Auxiliar.setIzq(Eliminar(Auxiliar.getIzq(),Dato));
105             }
106             else if (Dato.compareTo(Auxiliar.getElemento().getNomCompleto()) > 0)
107             {
108                 Auxiliar.setDer(Eliminar(Auxiliar.getDer(), Dato));
109             }
110             else if (Auxiliar.getIzq() != null && Auxiliar.getDer() != null)
111             {
112                 Auxiliar.setElemento(
113                             BuscarMayorIzquierda(Auxiliar.getIzq()).getElemento());
114                 Auxiliar.setIzq(EliminarMayorIzquierda(Auxiliar.getIzq()));
115             }
116             else
117                 Auxiliar = ( Auxiliar.getIzq() != null ) ?
118                             Auxiliar.getIzq() : Auxiliar.getDer();
119             return Auxiliar;
120         }
121     }
```

5. Diseñe el formulario **JFrame_Principal**





6. Implemente el código de la clase **JFrame_Principal**

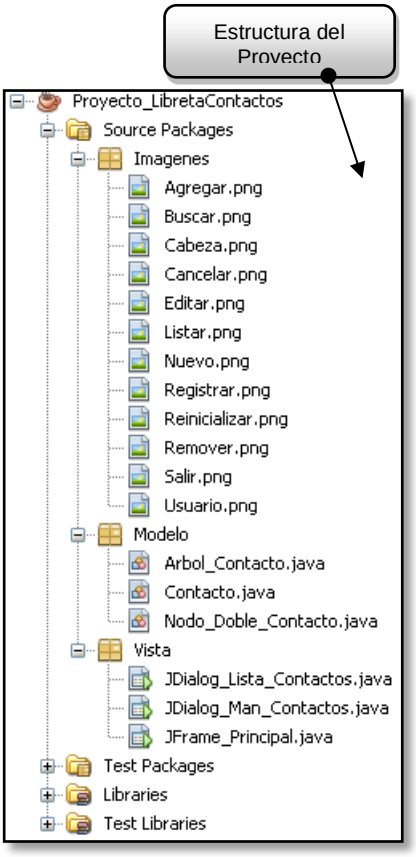
```java
package Vista;

import Control.Arbol_Cliente;
import Control.Nodo_Cliente;
import Modelo.Cliente;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class JFrame_Principal extends javax.swing.JFrame
{
    // 1. Cree los atributos necesarios
    Arbol_Cliente objArbol = new Arbol_Cliente(); // Instancia del Arbol
    DefaultTableModel ModTabla; // Modelo de la Tabla
    int Operacion; // Operacion (Nuevo o Editar)

    public JFrame_Principal()
    {
        initComponents();
        setLocationRelativeTo(null);
        /* 2. Llame al método Estado_Controles en (Estado 1) */
        Estado_Controles(false);
        /* 3. Asigne a la variable modTabla el modelo extraido
              del control Tbl_Datos */
        ModTabla = (DefaultTableModel) Tbl_Datos.getModel();
    }

    public void Limpiar_Controles()
    {
        /* 4. Implemente el método que borre el contenido de los controles
              Txt_Apellidos, Txt_Nombres y Txt_Telefono */
        Txt_Apellidos.setText("");
        Txt_Nombres.setText("");
        Txt_Telefono.setText("");
    }

    public final void Estado_Controles(boolean Estado)
    {
        /* 5. Implemente el método que realice el cambio de estado
              (Habilitar / Deshabilitar)
           Estado 1 --> (variable Estado == false)
           Estado 2 --> (variable Estado == true) */

        Txt_Apellidos.setEnabled(Estado);
        Txt_Nombres.setEnabled(Estado);
        Txt_Telefono.setEnabled(Estado);
        Btn_Nuevo.setEnabled(!Estado);
        Btn_Agregar.setEnabled(Estado);
        Btn_Cancelar.setEnabled(Estado);
        Btn_Buscar.setEnabled(!Estado);
        Btn_Eliminar.setEnabled(!Estado);
        Btn_Salir.setEnabled(!Estado);
        Tbl_Datos.setEnabled(!Estado);
    }
```

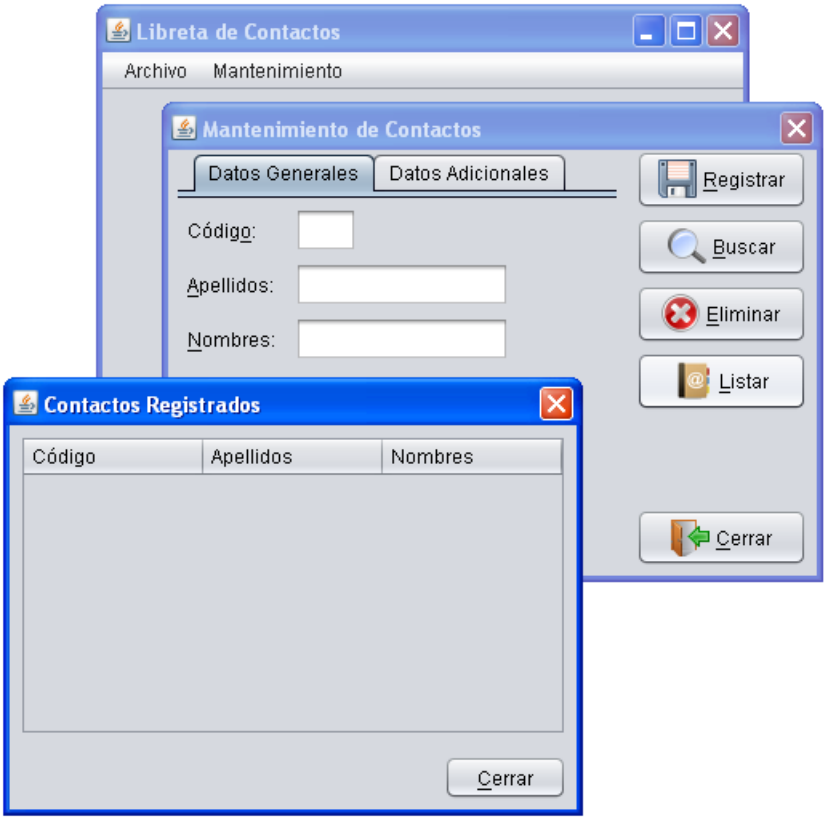
```
54
55      public void Limpiar_Tabla()
56      {
57          /* 6.Implemente el método que elimine todos los registros de la tabla */
58          ModTabla.setRowCount(0);
59      }
60
61      public void CargarDatos(Nodo_Cliente Auxiliar)
62      {
63          /* 7.Implemente el método que al recibir un nodo verifique que no
64               este vacio y carge el contenido del nodo en los controles */
65          if(Auxiliar != null)
66          {
67              Txt_Apellidos.setText(Auxiliar.getElemento().getApellidos());
68              Txt_Nombres.setText(Auxiliar.getElemento().getNombres());
69              Txt_Telefono.setText(Auxiliar.getElemento().getTelefono());
70          }
71      }
72
73      @SuppressWarnings("unchecked")
74      Generated Code
236
237     private void Btn_NuevoActionPerformed(java.awt.event.ActionEvent evt) {
238         // 8.Llame al método Estado_Controles en (Estado 2) y limpiar controles
239         Estado_Controles(true);
240         Limpiar_Controles();
241         // 9.Posicione el cursor en el control Txt_Apellidos
242         Txt_Apellidos.requestFocus();
243     }
244
245     private void Btn_AgregarActionPerformed(java.awt.event.ActionEvent evt) {
246         /* 10.Agregue los datos en un nodo del árbol.
247               Para ello deberá
248               (a) Crear un arreglo de objetos
249               (b) Crear una instancia de la clase Persona
250               (c) Llamar al método "Agregar" de la instancia Arbol enviando el
251                   nodo raiz y el elemento que será agregado  */
252         Object[] Registro = {Txt_Apellidos.getText(),
253                              Txt_Nombres.getText(),
254                              Txt_Telefono.getText()};
255         Cliente Elemento = new Cliente(Registro);
256         objArbol.setRaiz(objArbol.Agregar(objArbol.getRaiz(), Elemento));
257         // 11.Llame al método Limpiar_Tabla
258         Limpiar_Tabla();
259         /* 12.Actualice la tabla utilizando el método "Listar_InOrder".
260             No olvidar que al llamar a este método deberá de enviar el modelo
261             de la tabla y el nodo raiz  */
262         objArbol.Listar_InOrder(objArbol.getRaiz(), ModTabla);
263         // 13.Llame al método Limpiar_Controles
264         Limpiar_Controles();
265         // 14.Llame al método Estado_Controles en (Estado 1)
266         Estado_Controles(false);
267     }
```

```
268
269     private void Btn_CancelarActionPerformed(java.awt.event.ActionEvent evt) {
270         // 15.Llame al método Limpiar_Controles
271         Limpiar_Controles();
272         // 16.Llame al método Estado_Controles(Estado 1)
273         Estado_Controles(false);
274     }
275
276     private void Btn_BuscarActionPerformed(java.awt.event.ActionEvent evt) {
277         /* 17.Realice la operación de búsqueda. Para ello deberá:
278               (a) Solicite el nombre que desea buscar
279               (b) Declare una variable Auxiliar tipo Nodo que recibirá el
280                   resultado de la búsqueda realizada en el arbol
281               (c) Verificar el valor de la variable Auxiliar. Sólo si es
282                   diferente de null llamará al método Cargar_Datos, de lo
283                   contrario mostrará un mensaje indicando que el Nombre
284                   no existe  */
285         String Cadena = JOptionPane.showInputDialog("Nombre a Buscar: ");
286         Nodo_Cliente Auxiliar = objArbol.BuscarApeNom(Cadena);
287         if(Auxiliar != null)
288             CargarDatos(Auxiliar);
289         else
290             JOptionPane.showMessageDialog(this, "El Dato Buscado no existe");
291     }
292
293     private void Btn_EliminarActionPerformed(java.awt.event.ActionEvent evt) {
294         /* 18.Realice la operación de eliminación. Para ello deberá:
295               (a) Obtener el número de la fila seleccionada en la tabla
296               (b) Si el valor de la fila es diferente de -1 (que significa que
297                   tiene una fila seleccionada) continuará con la eliminación
298               (c) Cree una cadena compuesta de las dos primeras columnas de la
299                   fila seleccionada
300               (d) Llame al método "Eliminar" de la instancia Arbol enviando el
301                   nodo raiz y el elemento que será eliminado
302               (e) Llame al método Limpiar_Tabla
303               (f) Actualice la tabla utilizando el método "Listar_InOrder".
304                   No olvidar que al llamar a este método deberá de enviar
305                   el modelo de la tabla y el nodo raiz */
306         int Fila = Tbl_Datos.getSelectedRow();
307         if(Fila != -1)
308         {
309             String Cadena = ModTabla.getValueAt(Fila, 0) + " " +
310                             ModTabla.getValueAt(Fila, 1);
311             objArbol.setRaiz(objArbol.Eliminar(objArbol.getRaiz(),Cadena));
312             Limpiar_Tabla();
313             objArbol.Listar_InOrder(objArbol.getRaiz(), ModTabla);
314         }
315     }
316
317     private void Btn_SalirActionPerformed(java.awt.event.ActionEvent evt) {
318         // 19. Cerrar el Dialogo
319         dispose();
320     }
```

Ejercicio 2:

Elabore un proyecto que permita registrar dentro de una **Árbol** los datos (Código, Apellidos y Nombres) de un grupo de Contactos. El registro tomará como referencia al código del contacto y no deberá de permitir códigos repetidos. Asimismo, deberá de permitir realizar diversas operaciones como: Buscar, Eliminar y Listar.





Las Operaciones del Diálogo **"Mantenimiento de Contactos"** son:

a. **Registrar:**
   - Verifica que el código ingresado no esta repetido. Se utiliza el método **Buscar**
   - Sólo si el código no esta repetido se agregan los datos ingresados en los controles **JTextField** dentro de un nuevo **NODO** en el **ARBOL**

b. **Buscar:**
   - Solicita que se ingrese un Código en el control **TXT_CODIGO**
   - Busca el código en cada **NODO** del **ARBOL**
   - Si el código es encontrado en alguno de los **NODOS** se mostrarán en los controles **JTextField** todos los datos de dicho **NODO**

c. **Eliminar:**
   - Solicita que se ingrese un Código en el control **TXT_CODIGO**
   - Si el código existe dentro de algún **NODO** del **ARBOL**, dicho **NODO** será eliminado

d. **Listar:**
   - Carga el Diálogo del Listado de Contactos (Se pasará el **ARBOL** hacia ese diálogo) en donde se mostrarán en una tabla todos los contactos registrados.

e. **Cerrar:**
   - Cierra el diálogo.

Nota: El diseño del formulario es sólo referencial, lo importante es que se puedan realizar las operaciones solicitadas, es decir, el diseño del formulario lo puede desarrollar de forma diferente.