Manual Técnico

02/04/202

Introducción:

En el siguiente manual se explica cómo se llevo a cabo el proyecto. El proyecto hacia uso de diferentes estructuras no lineales, como la matriz dispersa y tres tipos de árboles. Se hizo referencia de los nodos que contenía cada estructura para que estas pudieran trabajar en conjunto y poder almacenar, buscar y eliminar datos de las estructuras que compartían información. Los árboles en relación con la búsqueda son eficientes, en algunos casos como el del árbol binario puede llegar a degenerarse si sus datos son ingresados en orden.

Paquetes:

- **Interfaz**
- **Matriz_Dispersa**
- **Árbol_b**
- **Estructuras**
- **Proyecto2_EDD**

Clases:

- **Iniciar_Sesión**
- **Navegación_imagenes**
- **Operaciones_usuarios**
- **Principal**
- **Registrar_Usuario**
- **Ventana_Usuario**
- **VisualizarEstructuras**
- **Encabezao**
- **ListaEnlazada**
- **Lista_2**
- **Matriz**
- **NodoEncabezado**
- **NodoListaDentro**
- **NodoLista_Enlazada**
- **Árbol_b**
- **Nodo**
- **Pagina**
- **Árbol_binario**
- **Atributos_arbolB**
- **Capas**
- **Cliente**
- **Colo_arbol**
- **ListaAvl**
- **Lista_Doble**
- **Lista_de_Album**
- **NodoAvl**
- **NodoArbolBinario**
- **NodoCiruclar**
- **NodoCola**
- **NodoPila**
- **NodoPixeles**

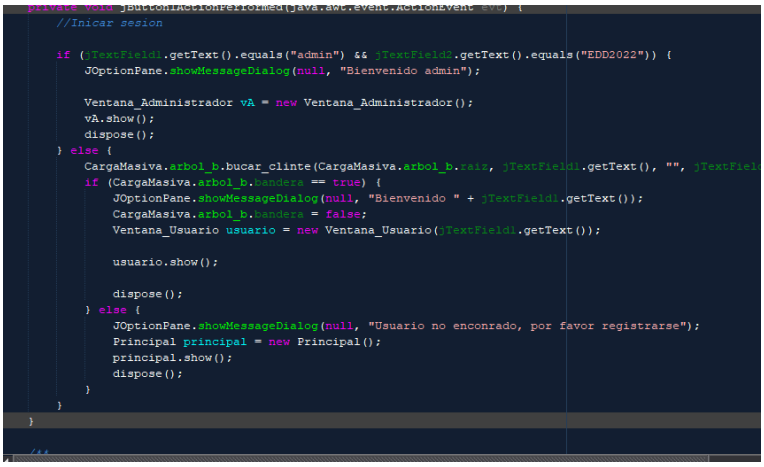
- **Nodo_Lista_Avl**
- **Pila**
- **Pixeles**
- **CargaMasiva**
- **Proyecto2_EDD**

Interfaz

En las clases que contiene este paquete se hace uso de drag and drop, para crear la parte grafica del proyecto.

Iniciar_Sesión:

En esta clase se validad que el usuario este registrado en el programa, para ello se tiene un usuario administrador el cual al acceder podrá agregar al resto de los usuarios.



```
    //Inicar sesion

    if (jTextField1.getText().equals("admin") && jTextField2.getText().equals("EDD2022")) {
        JOptionPane.showMessageDialog(null, "Bienvenido admin");

        Ventana_Administrador vA = new Ventana_Administrador();
        vA.show();
        dispose();
    } else {
        CargaMasiva.arbol_b.bucar_clinte(CargaMasiva.arbol_b.raiz, jTextField1.getText(), "", jTextFiel
        if (CargaMasiva.arbol_b.bandera == true) {
            JOptionPane.showMessageDialog(null, "Bienvenido " + jTextField1.getText());
            CargaMasiva.arbol_b.bandera = false;
            Ventana_Usuario usuario = new Ventana_Usuario(jTextField1.getText());

            usuario.show();

            dispose();
        } else {
            JOptionPane.showMessageDialog(null, "Usuario no enconrado, por favor registrarse");
            Principal principal = new Principal();
            principal.show();
            dispose();
        }
    }
}
```

Navegación_imagenes:

En esta parte el usuario, ya registrado podrá visualizar en el JFrame las imágenes generadas por los recorridos, ya se por capas usando el árbol binario o por imagene recorriendo por amplitud el árbol Avl. Se hacen las distintas validaciones para que el usuario pueda ingresar en los jText la cantidad de capas que de sea o el numero de la imagen.

```
    */
    public String id;

    public Navegacion_imagenes(String id) {
        this.id = id;
        initComponents();
    }

    public void mostrarPreorder(String id) {
        ImageIcon img = new ImageIcon(id);
        System.out.println("mostrar");
        JLabel label = new JLabel();

        label.setBounds(0, 0, img.getIconWidth(), img.getIconHeight());

        label.setIcon(img);
/         jScrollPane1.setViewportView(label);
    }

    public void mostrarInorder(String id) {
        ImageIcon img = new ImageIcon("MatrizIn_order.jpg");
        System.out.println("mostrar");
        JLabel label = new JLabel();

        label.setBounds(0, 0, img.getIconWidth(), img.getIconHeight());

        label.setIcon(img);
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    CargaMasiva.arbol_b.bucar_binario(CargaMasiva.arbol_b.raiz, id);
    Arbol_binario arbol_bi = CargaMasiva.arbol_b.arbol_bin;
    arbol_bi.cont = 0;
      arbol_bi.matriz = null;
    JLabel label = new JLabel();
    jLabel9.setText("");
    if (jComboBox1.getSelectedItem() == "Preorder") {
        arbol_bi.pre_orden(arbol_bi.raiz, Integer.parseInt(jTextField1.getText()));
        jLabel6.setText(arbol_bi.recorrido);
        ImageIcon img = new ImageIcon( new ImageIcon("MatrizPreOrder.jpg").getImage());

        label.setBounds(0, 50, img.getIconWidth(), img.getIconHeight());

        label.setIcon(img);
        jScrollPane1.setViewportView(label);
          this.mostrarPreorder("MatrizPreOrder.jpg");
        arbol_bi.matrizPreOden = null;
        arbol_bi.recorrido = "";
        arbol_bi.cont = 0;

    } else if (jComboBox1.getSelectedItem() == "Inorder") {
        arbol_bi.in_orden(arbol_bi.raiz, Integer.parseInt(jTextField1.getText()));
        jLabel6.setText(arbol_bi.recorrido);
        ImageIcon img = new ImageIcon( new ImageIcon("MatrizIn_order.jpg").getImage());
        System.out.println("mostrar");
```

Principal:

Es la clase principal de la interfaz gráfica, esta solo contiene dos botones, los cuales son el de registrar usuario y el de iniciar sesión.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    //Ventana registrar usuario

    Registrar_Usuario registrar = new Registrar_Usuario();
    registrar.show();
    dispose();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    Iniciar_Sesion iniciar_Sesion = new Iniciar_Sesion();
    iniciar_Sesion.show();
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Principal().setVisible(true);
```

Registrar_Usuarios:

En el JFrame de esta clase aparecen varios jText que el usuario debe de llenar si desea crear una cuenta. Se llama al método del árbol para insertar un nuevo cliente y se valida que esta no exista para posteriormente ser creado.

```
@SuppressWarnings("unchecked")
Generated Code

private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

//        JOptionPane.showMessageDialog(null, "Usuario creado con exito");
        Principal principal = new Principal();
        principal.show();
        dispose();

}

private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    CargaMasiva.arbol_b.insertar(Long.parseLong(jTextField3.getText()), jTextField2.getText(), jTextField
}

/**
```

Ventatana_Adiminsitrador

En esta clase se tiene un JFrame con todas las opciones que el administrador puede usar para poder manipular el programa. Ya sea cargar clientes, crearlos manualmente o eliminándolos.

```
ublic class Ventana_Administrador extends javax.swing.JFrame {

    /**
     * Creates new form Ventana_Administrador
     */
    JFileChooser jFile = new JFileChooser();
    File file;
    String ruta;

    public Ventana_Administrador() {
        initComponents();
    }

    public void mostrarArbol_B() {
        ImageIcon img = new ImageIcon( new ImageIcon("arbol_b.jpg").getImage());


        JLabel label = new JLabel();

        label.setBounds(0, 0, img.getIconWidth(), img.getIconHeight());

        label.setIcon(img);
        label.repaint();
        jScrollPanel.setViewportView(label);

    }
//////////
```

Ventana_Usuario

Esta clase es a la que accede el usuario cuando ya inicio sesión. Se tiene un JFrame y un JCombox con todas las opciones a las que puede acceder el usuario. Como la carga masiva de capas, imágenes y álbumes. También se tiene la opción que dirige al usuario a las clases Nvegacion_imagenes y Visualizacion de estructuras.

```
////////////////////////////////////////
    public String cargaImagenes() {
        String texto = "";
        if (jFile.showDialog(null, "Abrir") == JFileChooser.APPROVE_OPTION) {
            file = jFile.getSelectedFile();
            if (file.canRead()) {
                if (file.getName().endsWith("json")) {

                    ruta = file.getPath();
                    texto = CargaMasiva.abrirArchivo(file, ruta);

                } else {
                    JOptionPane.showMessageDialog(null, "Extensión erronea");

                }

            }
        }

        return texto;
    }
////////////////////////////////////////

    public String cargaCapas() {
        String texto = "";
```

Visualización de Estructuras:

Se tiene un JFrame, un jPanel, botones y un JScroll Pane. En el JScroll Pane se muestran las imágenes al presionar le boton.

```
        this.setLocationRelativeTo(null);
    }

//Mostrar árbol Abl
    public void mostrarArbol_Avl(String id) {
        ImageIcon img = new ImageIcon( new ImageIcon("arbol_avl" + id + ".jpg").getImage());

        JLabel label = new JLabel();

        label.setBounds(0, 0, img.getIconWidth(), img.getIconHeight());

        label.setIcon(img);
        jScrollPanel.setViewportView(label);
    }

    public void mostrarArbol_binario(String id) {
        ImageIcon img = new ImageIcon(new ImageIcon("ArbolBi" + id + ".jpg").getImage());

        JLabel label = new JLabel();

        label.setBounds(0, 0, img.getIconWidth(), img.getIconHeight());

        label.setIcon(img);
        jScrollPanel.setViewportView(label);
```

Matriz_Disperasa

Encabezado

En esta clase se tiene los método para crear los encabezado, tanto de la fila como de la columna.

```java
public class Encabezado {

   public  NodoEncabezado primero;

    public Encabezado(NodoEncabezado primero) {

        this.primero = primero;

    }

    public void setEncabezado(NodoEncabezado nuevo) {

        if (this.primero == null) {
            this.primero = nuevo;
        } else if (nuevo.getId() < this.primero.getId()) {
            nuevo.setSiguiente(this.primero);
            this.primero.setAnterior(nuevo);
            this.primero = nuevo;
        } else {

            NodoEncabezado actual = this.primero;
            while (actual.getSiguiente() != null) {
                if (nuevo.getId() < actual.getSiguiente().getId()) {
                    nuevo.setSiguiente(actual.getSiguiente());
                    actual.getSiguiente().setAnterior(nuevo);
                    nuevo.setAnterior(actual);
                    actual.setSiguiente(nuevo);
                    break;
```

Matriz:

En esta clase se tiene el método Insertar, el constructor de la matriz el cual tiene dos objetos de tipo encabezado uno para la fila y uno para la columna.

```java
public class Matriz {

    public Encabezado encabezadoFilas;
    public Encabezado encabezadoColumnas;

    public Matriz() {
        this.encabezadoColumnas = new Encabezado(null);
        this.encabezadoFilas = new Encabezado(null);
    }

    public void insertar(int fila, int columna, String valor, int cont) {
        Nodo nuevo = new Nodo(fila, columna, valor, cont);

        NodoEncabezado efilas = this.encabezadoFilas.getEncabezado(fila);

        if (efilas == null) {
            efilas = new NodoEncabezado(fila);
            efilas.setAcceso(nuevo);
            this.encabezadoFilas.setEncabezado(efilas);
        } else {

            if (nuevo.getColumna() < efilas.getAcceso().getColumna()) {
                nuevo.setDerecha(efilas.getAcceso());
```

```
    NodoEncabezado eColumnas = this.encabezadoColumnas.getEncabezado(columna);

    if (eColumnas == null) {
        eColumnas = new NodoEncabezado(columna);
        eColumnas.setAcceso(nuevo);
        this.encabezadoColumnas.setEncabezado(eColumnas);
    } else {
        if (nuevo.getFila() < eColumnas.getAcceso().getFila()) {
            nuevo.setAbajo(eColumnas.getAcceso());
            eColumnas.getAcceso().setArriba(nuevo);
            eColumnas.setAcceso(nuevo);

        } else {
            Nodo actual = eColumnas.getAcceso();
            while (actual.getAbajo() != null) {
                if (nuevo.getFila() < actual.getAbajo().getFila()) {
                    nuevo.setAbajo(actual.getAbajo());
                    actual.getAbajo().setArriba(nuevo);
                    nuevo.setArriba(actual);
```

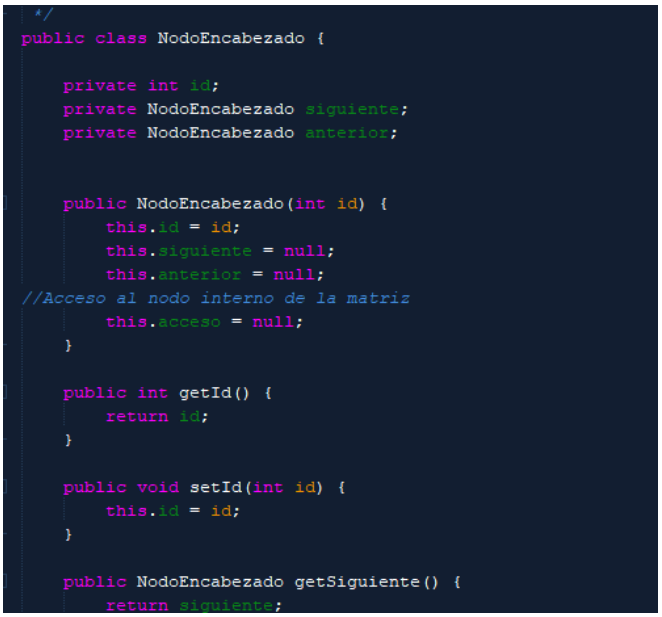
Función para graficar la matriz

En este método se crear un grafo para la matriz, el cual se guarda en una variable como una cadena para posteriormente retornarla. Se recorre primero la fila de la lista y después se recorre por columnas y así se van obteniendo los datos que contiene cada nodo de la matriz.

```
    public String grafico() {
//size='7.75,10.25'
        String principal = "digraph g{\n graph[size=\"5.75,5.25\"]\n label=\"Matriz dispersa\" \n node[s
        principal += "raiz[label=\"Inicio\",group=\"1\"]\nedge[dir=\"both\"]\n\n";
        int grupos;
        String F = "Fila";
        String C = "Columna";
        String N = "Nodo";
        Lista_2 listaFilas = new Lista_2();
        Lista_2 listaColumnas = new Lista_2();

        ListaEnlazada recorridoFilas = new ListaEnlazada();
        ListaEnlazada recorridoColumnas = new ListaEnlazada();

        NodoEncabezado eFila = this.encabezadoFilas.primero;
        NodoEncabezado eColumna = this.encabezadoColumnas.primero;

        while (eFila != null) {
            Nodo actual = eFila.getAcceso();

            listaFilas.add(actual);
//Se agrega a la lista los datos del nodo de la matriz.
//recorriendo por filas
            Lista_2 aux = new Lista_2();

            while (actual != null) {
                aux.add(actual);
                actual = actual.getDerecha();
            }
```

Nodo_Encabezado

Esta clase contiene los atributos de la clase Nodo_Encabezado, los cuales se usan en la lista de Encabezados para ir almacenando los datos como tipo Nodo_Encabezado junto con sus apuntadores siguiente y anterior.



```
 */
public class NodoEncabezado {

    private int id;
    private NodoEncabezado siguiente;
    private NodoEncabezado anterior;


    public NodoEncabezado(int id) {
        this.id = id;
        this.siguiente = null;
        this.anterior = null;
//Acceso al nodo interno de la matriz
        this.acceso = null;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public NodoEncabezado getSiguiente() {
        return siguiente;
```

Árbol_b

En esta clase se hace el método insertar para poder recibir y guardar los datos que corresponden a un cliente.

Métodos:

- **Insertar**
- **Insertar_en_pagina**
- **validarDivision**
- **dividir**
- **buscar_cliente**
- **buscar_cliente_**
- **buscar_binario**
- **buscar_binario_**
- **buscar_Avl_**
- **buscar_Avl**
- **buscar_ListaD**
- **buscar_ListaD**
- **modificar_cliente**
- **modificar_cliente_**
- **agregarArbolBinario**
- **ArgegarArbolBinario_**
- **Agregar_Arbol_Avl**
- **Agregar_Arbol_Avl_**
- **Agregar_Album**

- **Agregar_Albulm_**
- **crearGrafo**
- **getCodigos**

```java
public class Arbol_b {

    final int orden = 5;
    public Pagina raiz;
    public boolean bandera = false;
    public Arbol_binario arbol_bin = null;
    public Arbol_avl arbolAvl = null;
public Lista_Doble list_B=null;

    public Arbol_b() {
        this.raiz = new Pagina();

    }
//metodo insertar

    public void insertar(long id, String nombre, String password, Arbol_binario arbol_binario, Arbol_avl arbo
        Nodo nodo = new Nodo(id, nombre, password, arbol_binario, arbolAvl, listaD);
        Nodo objeto = insertar_en_pagina(nodo, raiz);
        if (objeto != null) {
            raiz = new Pagina();
            raiz.insertar(objeto);
            raiz.hoja = false;
        }

    }
```

```java
  private Nodo dividir(Pagina rama) {

      long valor = -999;
      String nombre = "";
      String password = "";
      Arbol_avl arbolAvl = null;
      Arbol_binario arbol_binario = null;
      Lista_Doble listaD = null;
      Nodo temp, nuevo;
      Nodo aux = rama.primero;

      Pagina rDerecha = new Pagina();
      Pagina rIzquierda = new Pagina();

      int contador = 0;
      while (aux != null) {
          contador++;

          if (contador < 3) {
              temp = new Nodo(aux.getId(), aux.getNombre_cliente(), aux.getPassword(), aux.getAr
              rIzquierda.hoja = !(temp.getRight() != null && temp.getLeft() != null);
              rIzquierda.insertar(temp);
          } else if (contador == 3) {
```

Nodo

Esta clase contiene los atributos que servirían para crear objetos de tipo Nodo en el Árbol b. Los cuales irán en las distintas páginas.

```java
public class Nodo {

    private long id;
    private String nombre_cliente;
    private String password;
    private Arbol_binario arbol_binario;
    private Arbol_avl arbolAvl;
    private Lista_Doble listaD;


    private Nodo anterior;
    private Nodo siguiente;
    private Pagina right;
    private Pagina left;

    public Nodo(long id, String nombre_cliente, String pasword, Arbol_binario arbol_binario, Arbol_avl arbo
        this.id = id;
        this.nombre_cliente = nombre_cliente;
```

Pagina

En esta clase se van validando los nodos e insertando en cada página.

```java
public class Pagina {
     boolean hoja;
    int cont;
   public Nodo primero;

    public Pagina() {
        this.primero = null;
        this.hoja = true;
        this.cont = 0;

    }

    public void insertar(Nodo nuevo) {

        if (primero == null) {

            primero = nuevo;
            this.cont++;
        } else {
            Nodo aux = this.primero;

            while (aux != null) {
                if ( nuevo.getId()==aux.getId()) {
                    System.out.println("El id" + nuevo.getId() + "ya existe");
                    break;
                } else {
```

Árbol_binario

En esta clase se hacen las validaciones para insertar nodos en un árbol binario de búsqueda, en el cual se valida que los valores que vayan ingresando no estén repetidos.

```
ublic class Arbol_binario {

    public NodoArbolBinario raiz;
    public Capas capa = null;
    public Matriz matriz = null;
    public boolean existeCapa = false;
    public Matriz matrizPreOden = new Matriz();
    public Matriz matrizInOrden = new Matriz();
    public Matriz matrizPost = new Matriz();
    public String recorrido = "";
    public int cont = 0;

    public Arbol_binario() {
        this.raiz = null;
    }

    public void agregar(Capas valor, Matriz matriz) {
        this.raiz = metodoRecursivo(valor, matriz, this.raiz);
    }

//Agregar
    public NodoArbolBinario metodoRecursivo(Capas valor, Matriz matriz, NodoArbolBinario raiz) {

        if (raiz == null) {
            NodoArbolBinario nodoArbol = new NodoArbolBinario(valor, matriz);
            return nodoArbol;

        } else {
            if (valor.getId_capa() < raiz.getCapas().getId_capa()) {
```

Capas

Esta clase tiene como función tener la estructura de una capa, la cual tiene como atributos un identificado y una lista de pixeles.

```java
 */
public class Capas {

    private int id_capa;
    private Pixeles pixeles;

    public Capas(int id_capa, Pixeles pixeles) {
        this.id_capa = id_capa;
        this.pixeles = pixeles;
    }

    public int getId_capa() {
        return id_capa;
    }

    public void setId_capa(int id_capa) {
        this.id_capa = id_capa;
    }

    public Pixeles getPixeles() {
        return pixeles;
    }

    public void setPixeles(Pixeles pixeles) {
        this.pixeles = pixeles;
    }
```

Cola

Esta cola tiene como función almacenar los datos de forma temporal que posteriormente son utilizados para poder graficar el árbol b.

```java
 * @author Kelly
 */
public class Cola_arbol_b {
     NodoCola frente;
   NodoCola fin;

    public Cola_arbol_b() {
        this.frente = null;
        this.fin = null;
    }

  public boolean colaVacia() {

        return this.frente == null;
    }

public void encolar(String padre,Pagina rama,int datos) {

        NodoCola nuevo = new NodoCola(new Atributos_arbolB(padre,rama, datos));
        if (colaVacia()) {
            this.frente = nuevo;

        } else {
            nuevo.setSiguiente(null);
```

Lista_Doble

Esta lista circular doblemente enlazada se utiliza para almacenar los datos de los álbumes. Cabe mencionar que esta lista almacena el nombre del álbum y una lista simplemente enlazada la cual va guardando las imágenes.

```
*/
public class Lista_Doble {

   public NodoCircular primero;
  public  NodoCircular ultimo;

    public Lista_Doble() {

        this.primero = null;
        this.ultimo = null;
    }

    public void insertar(String nombre, Lista_de_album lista) {

        NodoCircular nuevo = new NodoCircular(nombre, lista);

        if (this.primero == null) {
            this.primero = nuevo;
            this.primero.setSiguiente(this.primero);
            nuevo.setAnterior(this.ultimo);
            this.ultimo = nuevo;

//            this.primero = nuevo;
//            this.ultimo = nuevo;
//            this.primero.setSiguiente(this.ultimo);
//            this.primero.setAnterior(this.ultimo);
//            this.ultimo.setSiguiente(this.primero);
//            this.ultimo.setAnterior(this.primero);
        } else {
```

Lista_de_album

Esta lista sirve para almacenar las imágenes que pertenecen a un álbum.

```java
 * @author Kelly
 */
public class Lista_de_album {

    public Nodo_Album primero;
    public int tamanio;

    public Lista_de_album() {
        this.primero = null;
    }

    public void add(int imagen) {
        Nodo_Album nodo = new Nodo_Album(imagen);
        tamanio++;
        if (this.primero == null) {
            this.primero = nodo;
        } else {
            Nodo_Album aux = this.primero;
            while (aux.getSiguiente() != null) {
                aux = aux.getSiguiente();
            }
            aux.setSiguiente(nodo);
        }
    }

    public int get(int imagen) {
        Nodo_Album aux = this.primero;
```

NodoAvl

Esta clase contiene los atributo para crear los nodos del árbol Avl.

```java
 */
public class NodeAvl {

    private int dato;
    private Arbol_binario arbol_binaro;
    private int altura;
    private NodeAvl izquierdo;
    private NodeAvl derecho;

    public NodeAvl(int dato, Arbol_binario binario) {
        this.arbol_binaro = binario;
        this.dato = dato;
        this.altura = 1;
        this.izquierdo = null;
        this.derecho = null;
    }
```

NodoArbolBinario

Esta clase contiene los atributos para crear los nodos del árbol Avl.

```
 */
public class NodoArbolBinario {

    private Capas capas;
    private Matriz matriz;

    private NodoArbolBinario hijo_left;
    private NodoArbolBinario hijo_right;

    public NodoArbolBinario(Capas capas, Matriz matriz) {
        this.capas = capas;
        this.hijo_left = this.hijo_right;
        this.matriz = matriz;
    }
```

NodoCircular

Esta clase sirve para poder crear los Nodos de la lista circular doblemente enlazada.

```
public class NodoCircular {

    private String nombre;
    private Lista_de_album lista;
    private NodoCircular siguiente;
    private NodoCircular anterior;

    public NodoCircular(String nombre, Lista_de_album lista) {
        this.lista = lista;
        this.nombre = nombre;

        this.siguiente = null;
        this.anterior = null;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public NodoCircular getSiguiente() {
        return siguiente;
    }
```

NodoPila

Clase para crear nodos de tipo Pila.

```java
public class NodoPila {
    int dato;
    private Object nodo;
    private NodoPila siguiente;

    public NodoPila(Object nodo) {
        this.nodo = nodo;
        this.siguiente = null;
    }

    public Object getNodo() {
        return nodo;
    }

    public void setNodo(Object nodo) {
        this.nodo = nodo;
    }

    public NodoPila getSiguiente() {
        return siguiente;
    }
```

NodoPixeles

Nodo de la lista pixeles.

```java
public class NodoPixeles {

    private int fila;
    private int columna;
    private String color;
    private NodoPixeles siguiente;

    public NodoPixeles(int fila, int columna, String color) {
        this.fila = fila;
        this.columna = columna;
        this.color = color;
    }
```

Nodo_Album

Nodo de la lista álbum.

```java
 */
public class Nodo_Album {

    private int imagen;
    private Nodo_Album siguiente;

    public Nodo_Album(int imagen) {
        this.imagen = imagen;

        this.siguiente = null;
    }

    public Nodo_Album getSiguiente() {
        return siguiente;

    }

    public void setSiguiente(Nodo_Album siguiente) {
        this.siguiente = siguiente;
```

Pila

Esta pila se utiliza para las graficas guarda datos de tipo Object para poder ser usada en diferentes clases sin un tipo de dato definido.

```java
 * @author Kelly
 */
public class Pila {
        NodoPila cima;

    public Pila() {

        this.cima = null;

    }

    public boolean PilaVacia() {
        return this.cima == null;

    }

    public void push(Object nodoArbol) {
//String dato = null;
        //el nuevo nodo
        NodoPila nuevo = new NodoPila(nodoArbol);
        //el siguiente del nuevo nodo apuntara a la cima actual
        nuevo.setSiguiente(this.cima);
        //La cima cambia y se convierte en el nuevo nodo.
        this.cima = nuevo;

//        if (this.PilaVacia() == true) {
```

Pixeles

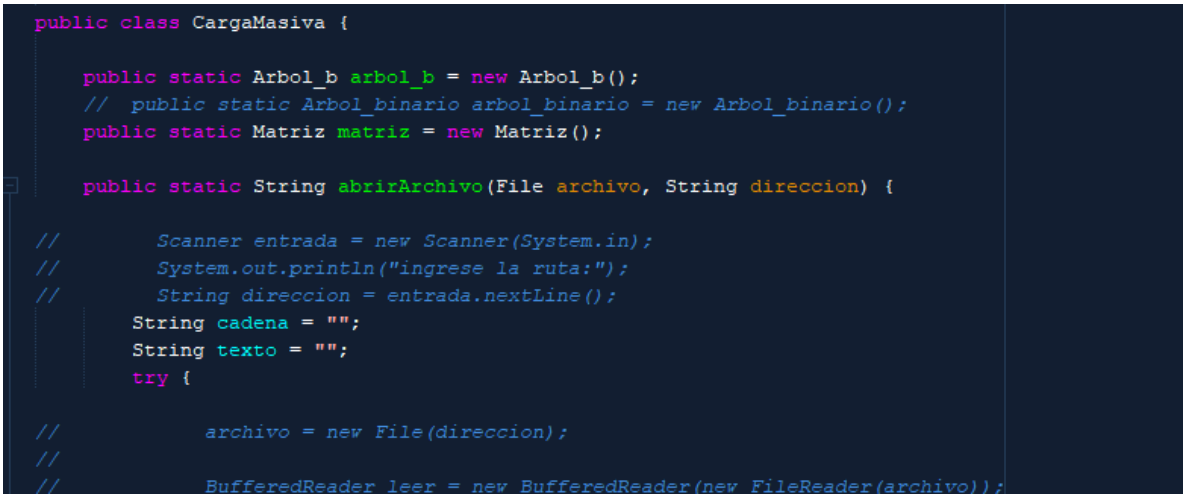
Lista de pixeles, en donde se almacenaran los mismos.

```
public class Pixeles {
public NodoPixeles primero;
   public int tamanio;

   public Pixeles() {
       this.primero = null;
       this.tamanio = 0;
   }

   public void add(int fila, int columna, String color) {
       NodoPixeles nodo = new NodoPixeles(fila, columna, color);
       tamanio++;
       if (this.primero == null) {
           this.primero = nodo;

       } else {
           NodoPixeles aux = this.primero;
           while (aux.getSiguiente() != null) {
               aux = aux.getSiguiente();
           }
           aux.setSiguiente(nodo);
       }
```

```
 */
public class Rama {

    int contador;
    boolean hoja;
    Nodo_b primero;

    public Rama() {
        this.contador = 0;
        this.hoja = true;
        this.primero = null;
    }

    void insertar(Nodo_b nodo) {

        if (this.primero == null) {
            this.primero = nodo;
            this.contador++;
        } else {
            Nodo_b temp = this.primero;
            do {
                if (nodo.dpi<=temp.dpi ) {
                    this.contador++;
                    if (temp == this.primero) {
                        this.primero.anterior = nodo;
                        this.primero.izquierda = nodo.derecha;
```

Carga masiva

En esta clase se extraen los datos de los distintos archivos utilizados durante la ejecución del programa.



```
public class CargaMasiva {

    public static Arbol_b arbol_b = new Arbol_b();
    //  public static Arbol_binario arbol_binario = new Arbol_binario();
    public static Matriz matriz = new Matriz();

    public static String abrirArchivo(File archivo, String direccion) {

//        Scanner entrada = new Scanner(System.in);
//        System.out.println("ingrese la ruta:");
//        String direccion = entrada.nextLine();
        String cadena = "";
        String texto = "";
        try {

//            archivo = new File(direccion);
//
//            BufferedReader leer = new BufferedReader(new FileReader(archivo));
```

Conclusión:

El programa se logró estructurar y acoplar a los requerimientos que se pedían por parte del cliente. Utilizando el paradigma de programación orientada a objetos se obtuvo una mejor organización del código que hacía más eficaz con una mejor organización a la hora de llamar clases y métodos, también se hizo uso de las distintas estructuras lineales para almacenar los datos de los clientes, imágenes y capas, que dicho usuario ingresaba por medio de un archivo json.