Universidad De San Carlos De Guatemala
Facultad De Ingeniería
Departamento de Ciencias Y Sistemas
Estructuras De Datos 1
Sección A
Ing. Jesús Alberto
Aux. Walter Oswaldo

MANUAL TÉCNICO
UDRAWING PAPER
Fase 2

Raudy David Cabrera Contreras
201901973

Manual Técnico

Clases Utilizadas:

- ❖ Recursos
- Cliente: para tener la estructura del cliente.
- Capa: para tener la estructura de las capas.
- Image: para tener la estructura de las imágenes.
- Album: para tener la estructura de los álbumes.
- ❖ Interfaz
- Admin: un jframe para la ventana del administrados.
- Inicio: un jframe para el inicio de sesión.
- Menu: un jframe para el menú principal del cliente.
- Modificar: un jframe para modificar los datos de un cliente.
- Registrar: un jframe para registrarse como cliente.
- ❖ Estructuras
- Lista: utilizada para la lista de álbumes.
- Lista Matriz: utilizada para las cabeceras.
- Matriz Dispersa: utilizada para graficar las capas
- Nodo Lista: Nodo para la lista de álbumes.
- Nodo Matriz: utilizada para el nodo de la matriz.
- ❖ Arboles
- Arbol ABB: utilizada para hacer la estructura de las capas.
- Arbol AVL: utilizada para hacer la estructura de las imágenes.
- Arbol B: utilizada para hacer la estructura de los clientes.
- Nodo AVL: utilizada para el nodo del árbol avl.
- Nodo ABB: utilizada para el nodo del árbol abb.
- Nodo B: utilizada para el nodo del árbol b.

Métodos

Clase **Cliente**

Los atributos de esta clase son el nombre, id del cliente, contraseña, lista de álbumes, árbol de capas, árbol de imágenes, cada uno con sus set’s y get’s.

```java
public class Cliente {

    private long dpi;
    private String nombre;
    private String pass;

    private List albumes;
    private ArbolAbb capas;
    private ArbolAvl imagenes;

    public Cliente(long dpi, String nombre, String pass) {
        this.dpi = dpi;
        this.nombre = nombre;
        this.pass = pass;
        albumes = new List();
        capas = new ArbolAbb();
        imagenes = new ArbolAvl();
    }
```

Cuenta con métodos como agregar álbum, contar los mismos, agregar capa y contar los mismos, agregar imagen y contar el mismo. Como sus métodos para graficar su álbum, su capa, y sus imágenes. Junto con sus reportes como el top 5, las capas de hojas, recorridos y profundidad. Donde cada uno regresa el grafico para generar la imagen.

```java
public void agregarAlbum(Album nAlbum){
    getAlbumes().insertar(nAlbum);
}

public int contarAlbumes(){
    return getAlbumes().tamanio();
}

public void agregarCapa(Capa capa){
    capas.insertar(capa);
}

public int contarCapas(){
    return capas.contarCapas();
}
```

```java
public void graficarImagenes(){
    imagenes.graficar(dpi+"_Imagenes");
    Main.graficarDot(dpi+"_Imagenes");
}

public void generarImagen(int n, String t){
    ArbolAbb capss = imagenes.buscar(n);
    MatrizDispersa img;
    img = capss.crearImagen(t);
    img.graficarMatriz(dpi+"_img"+n);
    Main.graficarDot(dpi+"_img"+n);
}
```

Clase **Image**

Los atributos son un id, size como cantidad, y un árbol abb para las capas, cada uno con sus respectivos set's y get's.

```java
public class Image {
    private int id;
    private int size;
    private ArbolAbb capas;

    public Image(int id, ArbolAbb capas) {
        this.id = id;
        this.capas = capas;
    }

    public Image(int id, int size) {
        this.id = id;
        this.size = size;
    }

    /**
     * @return the id
     */
```

Clase **Capa**

Los atributos son un id y una matriz dispersa para los pixeles de colores. Cada uno con sus set's y get's.

```java
public class Capa {
    private int id;
    private MatrizDispersa pixels;

    public Capa(int id, MatrizDispersa pixels) {
        this.id = id;
        this.pixels = pixels;
    }

    /**
     * @return the id
     */
    public int getId() {
        return id;
    }
```

Clase **Álbum**

Los atributos son un string para el nombre y una lista para las imágenes. Cuenta con sus respectivos set's y get's.

```java
public class Album {

    private String nombre;
    private List images;

    public Album(String nombre, List images) {
        this.nombre = nombre;
        this.images = images;
    }

    /**
     * @return the nombre
     */
    public String getNombre() {
        return nombre;
    }
```

Clase **Registrar**

Cuenta con un método para crear el cliente

```java
private void btnCrearActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(txtNombre.getText().equals("") || txtDpi.getText().equals("") || tx
        JOptionPane.showMessageDialog(null,"Ingrese Todos Los Campos","CL
    }else{
        try {
            long dpi = Long.parseLong(txtDpi.getText());
            String nombre = txtNombre.getText();
            String pass = txtPass.getText();
            Cliente nuevo = new Cliente(dpi,nombre,pass);
            Cliente prueba = Main.clientes.buscar(dpi);
            if(prueba==null){
                Main.clientes.insertar(nuevo);
                JOptionPane.showMessageDialog(null,"Usuario creado exitos
                Inicio in = new Inicio();
                in.setVisible(true);
```

Clase **Admin**

Cuenta con su método para cargar los clientes:

```java
private void btnCargaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int cont = 0;
    Scanner entrada = null;
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.showOpenDialog(fileChooser);
    String t = "";
    try {
        String ruta = fileChooser.getSelectedFile().getAbsolutePath();
        File f = new File(ruta);
        entrada = new Scanner(f);
        while (entrada.hasNext()) {
            if(cont==0){
                t += entrada.nextLine();
            }else{
```

Asi como su método para modificar a un cliente:

```
private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt)
    // TODO add your handling code here:
    String dp = JOptionPane.showInputDialog("Número de DPI:");
    try {
        long dpi = Long.parseLong(dp);
        Cliente objetivo = Main.clientes.buscar(dpi);
        if(objetivo != null){
            ModCliente mod = new ModCliente();
            mod.llenarlo(dpi, objetivo.getNombre(), objetivo.getPass());
            mod.setVisible(true);
        }else{
            JOptionPane.showMessageDialog(null, "DPI no encontrado.","ADMI
```

Como también el método para buscar un cliente:

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String dp = JOptionPane.showInputDialog("El número de DPI:");
    try {
        long dpi = Long.parseLong(dp);
        Cliente objetivo = Main.clientes.buscar(dpi);
        if(objetivo != null){
            verCliente(objetivo);
            try {
                Thread.sleep(1*1000);
```

También con el método para ver el árbol de los clientes:

```
private void btnVerArbolBActionPerformed(java.awt.event.ActionEvent evt)
    // TODO add your handling code here:
    Main.clientes.graficar("Clientes");
    try {
        Thread.sleep(2*1000);
    } catch (Exception e) {
        System.out.println(e);
    }
    ImageIcon imgg = new ImageIcon(System.getProperty("user.dir") + "\\Cl
    java.awt.Image imgfull = imgg.getImage().getScaledInstance(imagenMost
```

Así mismo un método para insertar clientes:

```
public static void addClientes(String text){
    JsonParser pr = new JsonParser();
    JsonArray arr = pr.parse(text).getAsJsonArray();
    for (JsonElement obj : arr) {
        JsonObject gsonObj = obj.getAsJsonObject();
        long dpi = Long.parseLong(gsonObj.get("dpi").getAsString(
        String nombre = gsonObj.get("nombre_cliente").getAsString
        String contra = gsonObj.get("password").getAsString();
        Cliente n = new Cliente(dpi,nombre,contra);
        Main.clientes.insertar(n);
```

Clase **Crear Cliente**

Cuenta con el método de crear el usuario una vez haya ingresado sus datos:

```
private void btnCrearActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(txtdpi.getText().equals("") || txtNombre.getText().equals("") ||
        JOptionPane.showMessageDialog(null,"Complete el Formulario.","E
    }else{
        try {
            long dpi = Long.parseLong(txtdpi.getText());
            String nombre = txtNombre.getText();
            String pass = txtPass.getText();
            Cliente nuevo = new Cliente(dpi,nombre,pass);
            Cliente obj = Main.clientes.buscar(dpi);
```

Clase **Inicio Sesión**

Ciuenta con el método para verificar los datos del usuario, si es admin o es cliente:

```
private void btnEntrarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String usuario = txtDpi.getText();
    String contra = txtPass.getText();

    if(usuario.equals("1999")){
        if(contra.equals("2022")){
            Admin admin = new Admin();
            admin.setVisible(true);
            dispose();
        }
    }else{
        long dpi = Long.parseLong(usuario);
```

Clase **Menu**

Es el menú para los usuarios, cuenta con la mayoría de funciones. Asi como el método para cargar los diferentes archivos:

```
JsonParser parser = new JsonParser();
JsonArray gsonArr = parser.parse(text).getAsJsonArray();
for (JsonElement obj : gsonArr) {
    JsonObject gsonObj = obj.getAsJsonObject();
    int id = gsonObj.get("id_capa").getAsInt();
    JsonArray pixeles = gsonObj.get("pixeles").getAsJsonArra
    MatrizDispersa mcapa = new MatrizDispersa();
    for(JsonElement pix: pixeles){
        JsonObject ps = pix.getAsJsonObject();
        int y  = ps.get("fila").getAsInt();
        int x  = ps.get("columna").getAsInt();
        String color = ps.get("color").getAsString();
        mcapa.insertar(color, x, y);
    }
    Capa nueva = new Capa(id,mcapa);
    String nombre = Main.actual.getDpi()+"_Capa"+id;
    mcapa.graficarMatriz(nombre);
```

También con los método de los reportes, donde se manda los datos para generar la imagen y luego mostrarla en pantalla:

```
private void btnTop5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Main.actual.generarTop5();
    String dpi = Main.actual.getDpi()+"_top5";
        try {
            Thread.sleep(2*1000);
        } catch (Exception e) {
            System.out.println(e);
        }
```

```
private void btnProfundidadActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Main.actual.profundidad();
    String id = Main.actual.getDpi()+"_profundidad";
        try {
            Thread.sleep(2*1000);
        } catch (Exception e) {
            System.out.println(e);
        }
    ImageIcon imgg = new ImageIcon(System.getProperty("user.dir") + "\\"+id+
```

Asi como el método para leer los json:

```
public static String leer(){
    int n = 0;
    Scanner entrada = null;
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.showOpenDialog(fileChooser);
    String t = "";
    try {
        String ruta = fileChooser.getSelectedFile().getAbsolutePath();
        File f = new File(ruta);
        entrada = new Scanner(f);
        while (entrada.hasNext()) {
```

Los analizadores para generar los diferentes archivos:

```
public static void analizarAlbumes(String text){
    JsonParser parser = new JsonParser();
    JsonArray gsonArr = parser.parse(text).getAsJsonArray();
    for (JsonElement obj : gsonArr) {
        JsonObject gsonObj = obj.getAsJsonObject();
        String nombre = gsonObj.get("nombre_album").getAsString();
        JsonArray imgs = gsonObj.get("imgs").getAsJsonArray();
        List albs = new List();
        for(JsonElement pix: imgs){
```

Clase **Nodo Lista**

Los atributos que tiene son objeto para información, dos tipo nodo, uno para el siguiente y otro para el anterior, con sus respectivos set's y get's.

```
public class NodoL {
    Object datoM;
    NodoL siguiente;
    NodoL anterior;

    public NodoL(Object datoM) {
        datoM = datoM;
        siguiente = null;
        anterior = null;
    }

}
```

Clase **Nodo Matriz**

Tiene los atributos de información, nodos para abajo, arriba, derecha, izquierda, siguiente y anterior, como posiciones en i y j:

```
Object info;
Nodo izquierda, derecha, arriba, abajo;
Nodo sig, ant;
int x,y;

public Nodo(Object info) {
    this.info = info;
    x = y = 0;
    this.sig= null;
    this.ant = null;
    this.izquierda = null;
    this.derecha = null;
    this.abajo = null;
    this.arriba = null;
```

Contiene sus métodos de insertar;

```
public void insertar(String valor, int i, int j) {
    Nodo fil = fila.buscarList(i);
    Nodo com = col.buscarList(j);
    String nuevoColor = buscar_Color(i, j);
    if (nuevoColor.equals("")) {
        if (fil == null && com == null) {
            //No existe Fila ni Columna
            casol(valor, i, j);
        } else if (fil == null && com != null) {
            //Solo existe Col
```

Asi como para agregar capas pasando la matriz como parámetro:

```
public void agregarCapa(MatrizDispersa nueva){
    if(nueva != null){
        Nodo cabecera = nueva.fila.raiz;
        while(cabecera != null){
            Nodo aux = cabecera.abajo;
            while(aux != null){
                String color = buscar_Color(aux.x,aux.y);
                if(color.equals("")){
                    this.insertar((String)aux.info,aux.x,aux.y);
                }else{
```

Clase **Arbol ABB**

Cuenta con el atributo de raíz del tipo de su nodo.

```java
public class ArbolAbb {
    NodoAbb raiz;

    public ArbolAbb() {
        raiz = null;
    }
```

Asi como insertar elementos en el:

```java
if(raiz == null){
    return new NodoAbb(inf);
}else{
    int id = ((Capa)inf).getId();
    int actual = ((Capa)raiz.info).getId();
    if(id < actual){
        raiz.izquierda = addRecur(inf,raiz.izquierda);
    }else if(actual < id){
        raiz.derecha = addRecur(inf,raiz.derecha);
    }
    return raiz;
}
```

El método buscar dentro de el:

```java
public MatrizDispersa buscar(int n){
    NodoAbb aux = raiz;
    while(aux != null){
        if(((Capa)aux.info).getId() == n){
            return((Capa)aux.info).getPixels
        }else if(n<((Capa)aux.info).getId())
            aux = aux.izquierda;
        }else{
            aux = aux.derecha;
```

Clase **Arbol Avl**

Cuenta con los métodos para insertar dentro de el:

```java
public void insertar(Object info){
   NodoAVL nuevo= new NodoAVL(info);
   if(raiz==null){
      raiz=nuevo;
   }else{
      raiz=insertar(nuevo, raiz);
   }
```

El factor para saber el equilibro que tiene.

```java
public int factorEquilibrio(NodoAVL n){
    if(n==null){
        return -1;
    }else{
        return n.equilibrio;
    }
}
```

Los método de rotación:

```java
public NodoAVL rotacionDobleDerecha(NodoAVL nodo){
    NodoAVL aux;
    nodo.derecha= rotacionIzquierda(nodo.derecha);
    aux=rotacionDerecha(nodo);
    return aux;
}
```

```java
public NodoAVL rotacionDobleIzquierda(NodoAVL nodo){
    NodoAVL aux;
    nodo.izquierda= rotacionDerecha(nodo.izquierda);
    aux=rotacionIzquierda(nodo);
    return aux;
}
```

Eliminar del árbol:

```
public NodoAVL eliminar(int n, NodoAVL raiz){
    if(((Image)raiz.valor).getId() == n){
        if(raiz.izquierda == null && raiz.derecha == nu
            raiz = null;
        }else if(raiz.izquierda != null){
            Image x = obtenerMayor(raiz.izquierda);
            raiz.valor = x;
            raiz.izquierda = eliminar(x.getId(),raiz.iz
        }else{
            raiz = raiz.derecha;
        }
    }else{
        if(((Image)raiz.valor).getId()<n){
```

Clase **Árbol B**

Cuenta con su método para insertar:

```
public void insertar(Cliente cl){
    Cliente n = brecursivo(cl.getDpi(),raiz);
    if(n!=null){
        JOptionPane.showMessageDialog(null, "El Cliente con DPI: "+n.getI
    }else{
        NodoB aur = raiz;
        insertarLL(raiz,cl);
        if(lleno(raiz)){
            NodoB aus = new NodoB();
            aus.hoja = false;
            raiz = aus;
            raiz.n0 = aur;
```

Buscar dentro de el:

```
  NodoB aux = raiz;
  if(aux.info1 != null && aux.info1.getDpi() == dpi){
      return aux.info1;
  }else if( aux.info2 != null && aux.info2.getDpi() == dpi){
      return aux.info2;
  }else if(aux.info3 != null && aux.info3.getDpi() == dpi){
      return aux.info3;
  }else if(aux.info4 != null && aux.info4.getDpi() == dpi){
      return aux.info4;
  }else{
      if(hijos(aux)){
          if(aux.info1 == null || (aux.n0 != null && aux.info1.getDpi
```

Saber si tiene hijos:

```
public boolean hijos(NodoB nod){
    if(nod.n0 != null || nod.n1 != null || nod.n2 != null || n
        return true;
    }else{
        return false;
    }
```

Clase **Nodo Abb**

Tiene como atributos un valor como información, dos nodos del mismo com derecha e izquierda:

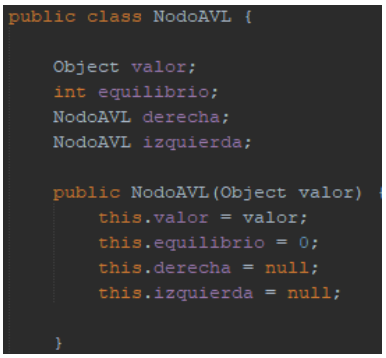
```
public class NodoAbb {

    Object info;
    NodoAbb derecha;
    NodoAbb izquierda;

    public NodoAbb(Object info) {
        this.info = info;
        this.derecha = null;
        this.izquierda = null;
    }
}
```

Clase **Nodo Avl**

Tiene como atributos un valor, un entero para el equilibro y dos nodos del mismo para la derecha e izquierda:



```java
public class NodoAVL {

    Object valor;
    int equilibrio;
    NodoAVL derecha;
    NodoAVL izquierda;

    public NodoAVL(Object valor) {
        this.valor = valor;
        this.equilibrio = 0;
        this.derecha = null;
        this.izquierda = null;

    }
```