Universidad de San Carlos de Guatemala
Facultad de ingeniería
Ingeniería en Ciencias y Sistemas
LENGUAJES FORMALES Y DE PROGRAMACION

MANUAL TECNICO

Alberto Josue Hernández Armas 201903553

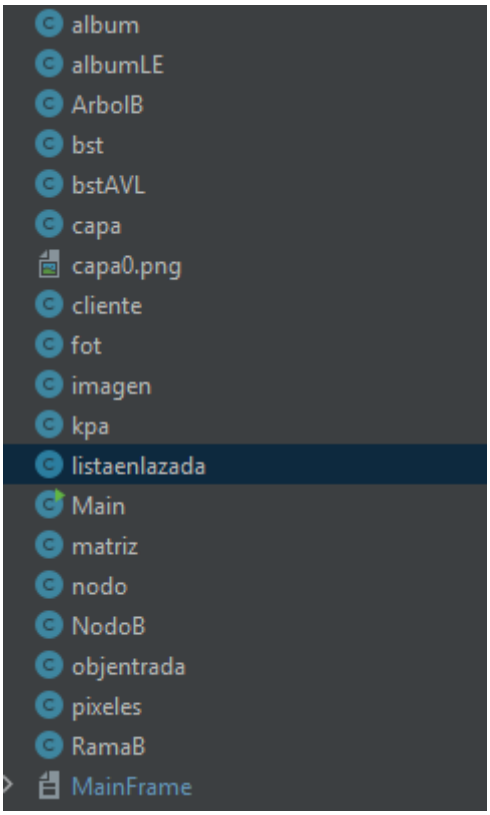
Guatemala 2 de Abril 2022

INTRODUCCION

En el presente manual técnico podremos observar cómo es que está diseñado el código y explicando que forman la funcionalidad de una manera general, teniendo como objetivo principal que sea más entendible. Determinando los métodos utilizados y explicando algunas palabras claves refiriéndonos a sus propiedades demostrando cuál es su función dentro de los bloques de código que se nos presentan a continuación.

MANUAL TECNICO

Se cuenta con las siguiente clases y objetos para la realización correcta de este proyecto

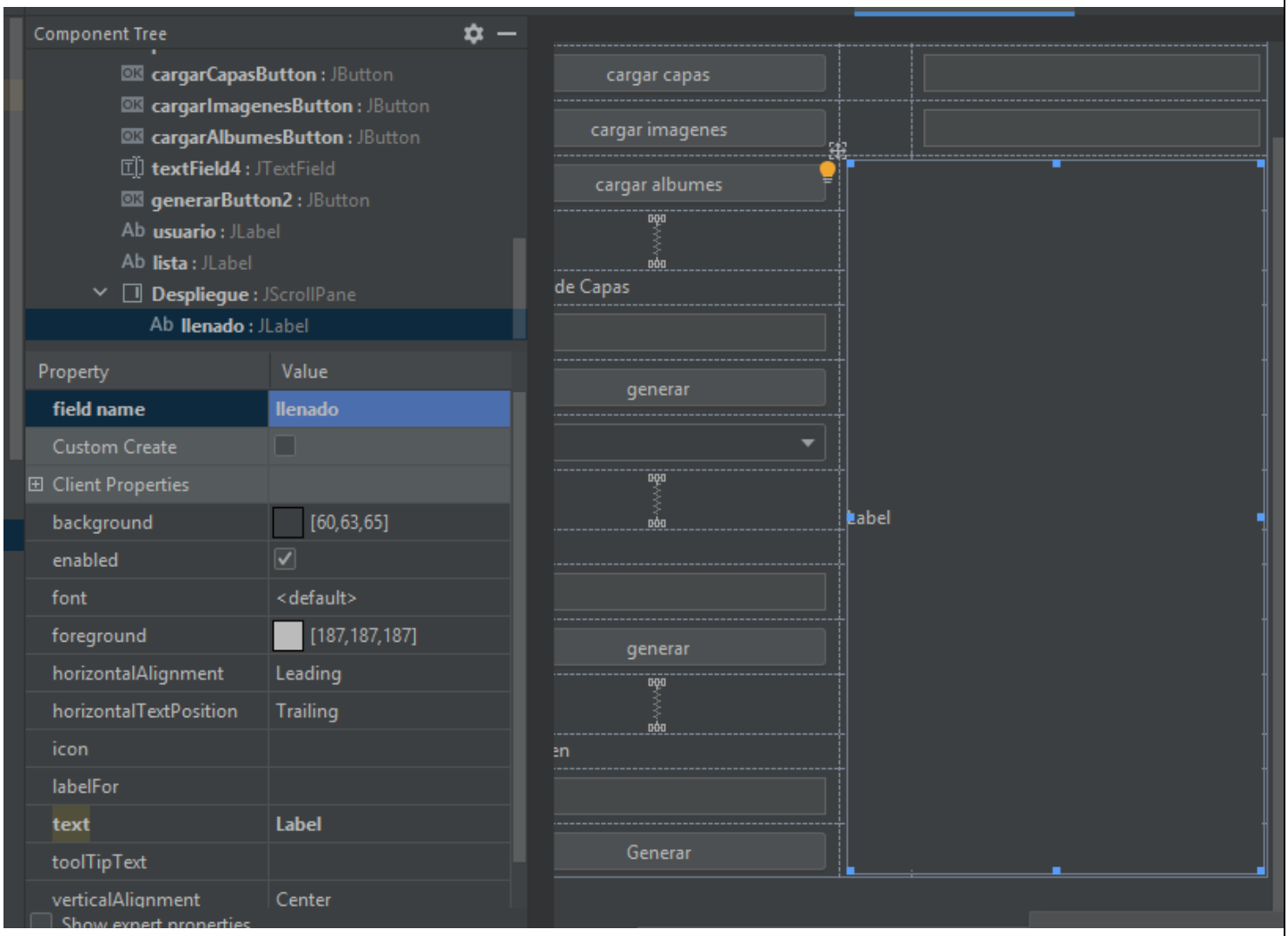


```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import com.google.gson.*;

import java.io.*;
import java.security.PublicKey;
import java.util.Set;
import java.util.HashSet;
import java.io.FileWriter;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.security.Principal;
import java.util.Set;
import java.util.HashSet;

import java.io.FileReader;
import java.security.Key;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Iterator;
import java.util.Scanner;
import javax.print.DocFlavor;
import javax.swing.*;
```

Primero podemos observar el diseño de la interfaz grafica, la cual cuenta con los distintos espacios para el tratamiento de la información que entrara.



Luego podemos observar el funcionamiento del motor principal de la aplicación.

```
//EspacioImagenes.setVisible(true);
clientes = new ArbolB();
clientesemg = new listaenlazada();
cliente administrador = new cliente( dpi: "3004548210101", nombre_cliente: "Alberto", password: "zapata");
nodo primero = new nodo(administrador);
clientesemg.agrega(primero);



cliente admin = new cliente( dpi: "3004548210101", nombre_cliente: "Admin", password: "EDD2022");
clientes.insertar(admin);

setContentPane(MainPanel);
setTitle("Hola como tas");
setSize( width: 700, height: 700);
comboBox1.addItem("preorder");
comboBox1.addItem("inorder");
comboBox1.addItem("postorder");
setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
setVisible(true);
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                ((cliente)cliente_actual.value).generarcapas();
                System.out.println(((cliente)cliente_actual.value).arbol_capas);
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    });
    cargarImagenesButton.addActionListener(new ActionListener() {...});
    cargarAlbumesButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                ((cliente)cliente_actual.value).generaLE();
                System.out.println(((cliente)cliente_actual.value).avl_imagenes);
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    });
    logINButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
```

Por motivos de ahorro de código se implemento lo que es el polimorfismo en la mayoría de las clases utilizadas

```
    public   Object value;
    public int blnc;
    public   nodo izq, der,uniq,Next,Prev,abajo,ladod;
    public String propiedad;
    public int fila, columna;
    public nodo(Object value){
        this.value = value;
        this.izq = null;
        this.der = null;
        this.abajo = null;
        this.ladod = null;
        this.Next = null;
        this.Prev = null;
        this.fila = -1;
        this.columna = -1;
    }
    public void agregabnc(int balance)
    {
        this.blnc = balance;

    }

    public void  agregaderecha(Object valor)
    {
        nodo mnodo = new nodo(valor);
        if(this.der==null)
```

Bst:

```
public nodo root;
public static String texto;
public bst() { this.root = null; }
public void insert(Object valor)
{...}
public  static void preordercorreccion(nodo n)
{
    if (n == null)
        return;
    ((kpa)n.value).m.correcion();
    ((kpa)n.value).printpng();//ACTIVARLO PARA VOLVER A IMPRIMIR LOS

    System.out.println(n.value);

    preordercorreccion(n.izq);
    preordercorreccion(n.der);


}
public void correccion() { preordercorreccion(this.root); }
public void _insert(Object valor, nodo ahora)
{...}
public static void imprime_arbol(nodo root)
{...}


@Override
public String toString() {...}
```

AVL:

```java
package com.company;
//_INSERT CAMBIAR EL VALOR DE COMPARACION
public class bstAVL {
    public nodo root;
    public nodo va;
    public int result = 0;
    public static String texto;
    public static nodo desbalanceado;
    public listaenlazada desbalanceados = new listaenlazada();

    public bstAVL() { this.root = null; }
    public int contizq=0, contder = 0;
    public void insert(Object valor)
    {...}
    public void verificarizq(nodo root, nodo referente)
    {...}
    public void verificarder(nodo root, nodo referente)
    {...}
    public void preordercorreccionbalanceo(nodo n)
    {
        if (n == null)
            return;
        System.out.println(n.value);
        boolean s = this.verificaravl(n);
        System.out.println(s);
        if (!s) {
            //desbalanceado = n;
            //va = desbalanceado;
```

```
public static void imprime_arbol(nodo root)
{...}
public void balanceoavl()
{
    System.out.println("EL PROBLEMA ES DE BALANCEA");
    nodo ahora = null;
    nodo tmp = this.desbalanceados.Last;
    if(tmp.blnc < 0) ahora = tmp.izq;
    else if(tmp.blnc > 0) ahora = tmp.der;
    else  ahora = null;
    ahora.Prev = null;
    if(tmp == this.root)
    {
        this.root = ahora;
        System.out.println("SI ENTRO HASTA ACA EN EL THIS.ROOT = AHORA"+this.root);
    }
    else
    {
        if(tmp.Prev.der != null && tmp.Prev.der == tmp) {
            tmp.Prev.der = ahora;
            ahora.Prev = tmp.Prev;
        }
        else if(tmp.Prev.izq != null && tmp.Prev.izq == tmp) {
            tmp.Prev.izq = ahora;
            ahora.Prev = tmp.Prev;
        }
```

ARBOL B:

```
    int orden_arbol = 5;
    RamaB raiz;

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

    public void insertar(cliente id) {
        NodoB nodo = new NodoB(id);
        if (raiz == null) {
            raiz = new RamaB();
            raiz.insertar(nodo);
        } else {
            NodoB obj = insertar_en_rama(nodo, raiz);
            if (obj != null) {
                //si devuelve algo el metodo de insertar en rama quiere
                raiz = new RamaB();
                raiz.insertar(obj);
                raiz.hoja = false;
            }
        }
    }

    private NodoB insertar_en_rama(NodoB nodo, RamaB rama) {
        if (rama.hoja) {
            rama.insertar(nodo);
            if (rama.contador == orden_arbol) {
                //si ya se insertaron todos los elementos posibles se d
```

```
private NodoB dividir(RamaB rama) {
    int val = -999;
    NodoB temp, Nuevito;
    NodoB aux = rama.primero;
    RamaB rderecha = new RamaB();
    RamaB rizquierda = new RamaB();

    int cont = 0;
    while (aux != null) {
        cont++;
        //implementacion para dividir unicamente ramas de 4 nodos
        if (cont < 3) {
            temp = new NodoB(aux.id);
            temp.izquierda = aux.izquierda;
            if (cont == 2) {
                temp.derecha = aux.siguiente.izquierda;
            } else {
                temp.derecha = aux.derecha;
            }
            //si la rama posee ramas deja de ser hoja
            if (temp.derecha != null && temp.izquierda != null) {
                rizquierda.hoja = false;
```

Cada una de estas estructuras tiene un valor String, al hacer llamado a ese valor, genera un código graphviz para ver el estado de memoria de manerea grafica, en este punto también se implemento polimorfismo.