Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Área de Desarrollo de Software
Laboratorio Estructura de Datos

Proyecto 2

UDrawing Paper

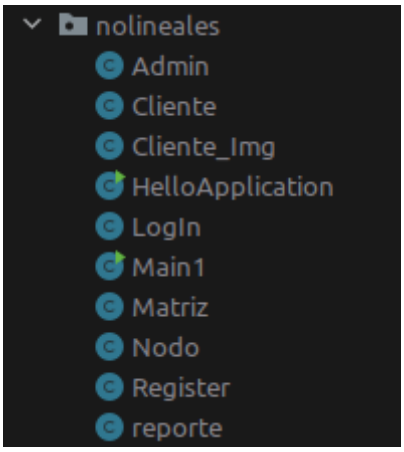
Implementación de estructuras no lineales

Nombre: Eddy Fernando Díaz Galindo
Registro académico: 201906558
Instructor: Alex Lopez
Fecha de Entrega: 02/04/2022

CLASES DE UDRAWING PAPER

Clase	Descripción
Admin	Clase encargada de mostrar el contenido del usuario administrador del sistema.
Cliente	Clase encargada de mostrar las capas generadas por el usuario cliente del sistema.
Cliente_img	Clase encargada de mostrar las imagenes generadas por el usuario cliente del sistema. Realizar recorridos para generar imágenes o simplemente mostrar una o mas capas apiladas
HelloApplication	Es la clase sub-principal del sistema ya que contiene el contenido grafico del sistema y los cambios entre ventanas y clases.
Login	Clase encargada de mostrar el login y de validar las credenciales que ingresa el usuario.
Main 1	Clase principal encargada de iniciar el programa para que ejecute el jar
Matriz	Clase encargada de crear matrices que contengan nodos con la información del pixel a dibujar
Nodo	Clase que genera los nodos de la clase matriz
Register	Clase encargada de controlar los usuario que desean registrarse en el sistema.
reporte	Clase encargada de realizar los reportes de cada capa, imagen o estructura.

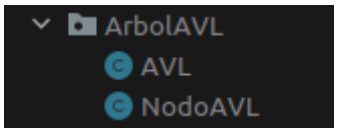
Fuente: Elaboración Propia, 2022



Fuente: Elaboración Propia, 2022

Clase	Descripción
AVL	Clase encargada de insertar imágenes al arbol AVL y generar reportes
NodoAVL	Clase encargada de generar nodos de la clase AVL

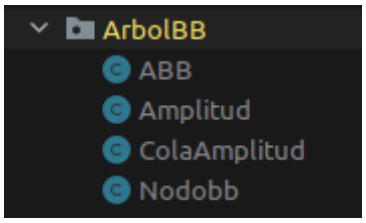
Fuente: Elaboración Propia, 2022



Fuente: Elaboración Propia, 2022

Clase	Descripción
ABB	Clase encargada de insertar capas al arbol ABB y generar reportes
Amplitud	Clase cola encargada de guardar los nodos en espera que pasaran para generar la imagenen por recorrido de amplitud.
ColaAmplitud	Clase nodo encargada de guardar el contenido de los nodos del recorrido por amplitud
Nodobb	Clase nodo encargado de guardar los nodos del arbol binario de busqueda que contiene las capas.

Fuente: Elaboración Propia, 2022



Fuente: Elaboración Propia, 2022

MÉTODOS PRINCIPALES DE UDRAWING PAPER

Método Start: Método encargado de iniciar la interfaz gráfica. Existen mas métodos similares a este ya que hay mas ventanas como la carga de capas que tambien necesitan ser inciadas.

```
public class HelloApplication extends Application {

    public static Stage stg;
    public static Stage StageRegiste;
    public static Stage StageCliente;
    public static Stage StageImag;
    public static Stage StageReport;

    public static Scene scene;
    public static boolean cambioCapa = false;
    public static boolean cambioImag = false;
    public static boolean cambioReport= false;

    public static boolean cambioRegister = false;
    public static boolean cambioAdmin = false;

    @Override
    public void start(Stage stage) throws IOException {
            FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource( name: "hello-view.fxml"));
            scene = new Scene(fxmlLoader.load(),  v: 600,  v1: 400);
            stage.setTitle("[EDD]Fase2!");
            stage.setScene(scene);
            stage.getIcons().add(new Image(String.valueOf(getClass().getClassLoader().getResource( name: "logo.png"))));
            stage.show();
            stg = stage;
    }
```

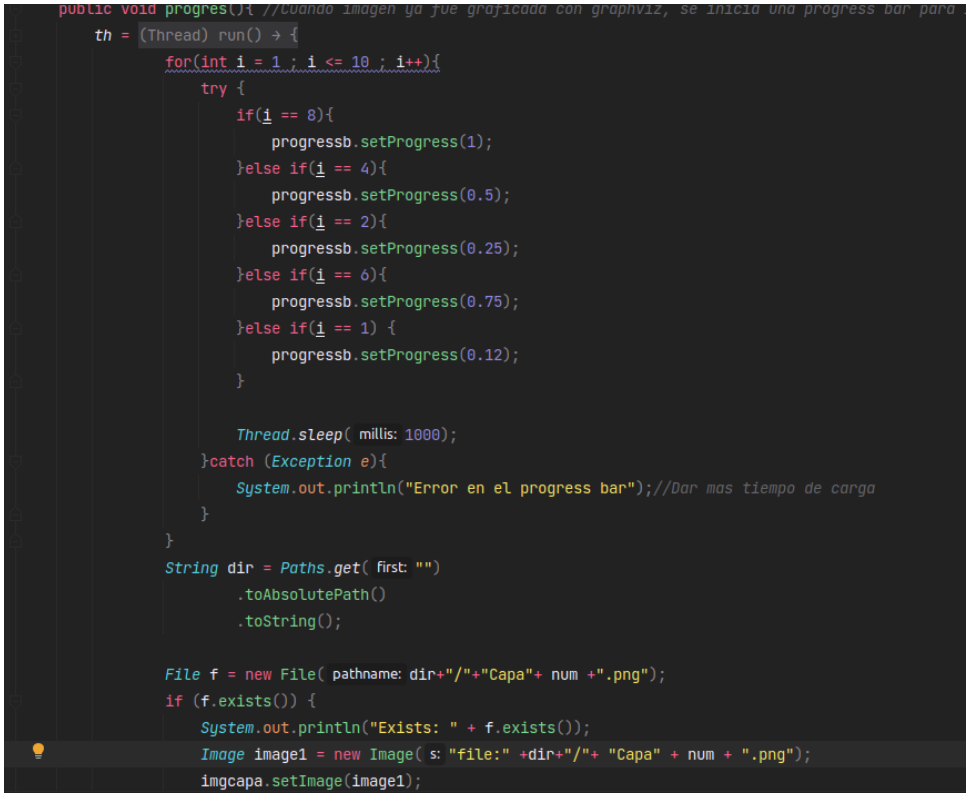
Fuente: Elaboración Propia, 2022

Método RutaArchivo: Método encargado de obtener la ruta del archivo json que desea ser cargado al sistema. Este método pasa por parámetro al método Analyzer que es el encargado de obtener y separar el contenido del json.

```java
public void RutaArchivo() throws IOException {
    cmbx.getItems().clear();
    String ruta = "";
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Cargar el archivo");
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter( s: "JSON",  ...strings: "*.json"),
            new FileChooser.ExtensionFilter( s: "ALL FILES",  ...strings: "*.*")
    );

    File file = fileChooser.showOpenDialog(new Stage());
    if (file != null) {
        ruta = file.getPath();
        Analyzer(ruta);
    } else {
        System.out.println("Error, seleccione json");
    }
}

public void Analyzer(String path) {
    int fila;
    int columna;
    try {
        JSONParser parser = new JSONParser();
        Object obj = parser.parse(new FileReader(path));
        JSONArray jsonarra = (JSONArray) obj;

        for (int i = 0; i < jsonarra.size(); i++) {
            JSONObject jsonobj1 = (JSONObject) jsonarra.get(i);
```

Fuente: Elaboración Propia, 2022

Método progres: Es el encargado de detener el programa para que busque y cargue la imagen o capa generada en los archivos de nuestro sistema.



```
public void progres(){ //Cuando imagen ya fue graficada con graphviz, se inicia una progress bar para
    th = (Thread) run() -> {
            for(int i = 1 ; i <= 10 ; i++){
                try {
                    if(i == 8){
                        progressb.setProgress(1);
                    }else if(i == 4){
                        progressb.setProgress(0.5);
                    }else if(i == 2){
                        progressb.setProgress(0.25);
                    }else if(i == 6){
                        progressb.setProgress(0.75);
                    }else if(i == 1) {
                        progressb.setProgress(0.12);
                    }

                    Thread.sleep( millis: 1000);
                }catch (Exception e){
                    System.out.println("Error en el progress bar");//Dar mas tiempo de carga
                }
            }
            String dir = Paths.get( first: "")
                    .toAbsolutePath()
                    .toString();

            File f = new File( pathname: dir+"/"+"Capa"+ num +".png");
            if (f.exists()) {
                System.out.println("Exists: " + f.exists());
                Image image1 = new Image( s: "file:" +dir+"/"+ "Capa" + num + ".png");
                imgcapa.setImage(image1);
```

Fuente: Elaboración Propia, 2022

Método CrearImgPAI: Este método es el encargado de generar imágenes por recorrido en amplitud, crea una conexión entre el arbol avl y las capas del arbol binario. Esta ligado al método progres

```
public void CrearImgPAI(ActionEvent actionEvent) {
    reiniciarimg();
    num = "";
    String contxbox;
    contxbox = txboxnum.getText();

    try{
        NodoAVL conexion = arbolavl.IniciarBusqueda(Integer.parseInt(contxbox));
        conexion.InicioAmplitud();
        String[] ids1 = conexion.contAmplitud.split( regex: ",");
        for(int i = 0; i < ids1.length;i++) {
            num += ids1[i] + " ";
            Nodobb conexion3 = x.IniciarBusquedabb(Integer.parseInt(ids1[i]));
            nuevaImagen(conexion3);
        }
    }catch (Exception e){
        System.out.println("ID no encontrado");
    }
    imagenes.mayorcol();
    imagenes.mayorfila();
    imagenes.graficar(num);
    progres();
}
```

Fuente: Elaboración Propia, 2022

Método checkLogin: Este método es el encargado de validar los credenciales de los usuarios.

```
private void checkLogin() throws IOException {

    if(txuser.getText().toString().equals("admin") && txpass.getText().toString().equals("123")) {
        lbwrong.setText("admin!");
        m.changeAdmin();
    }else if(txuser.getText().toString().equals("cliente") && txpass.getText().toString().equals("123")) {
        lbwrong.setText("cliente!");
        m.changeCliente();
    }else if(txuser.getText().isEmpty() && txpass.getText().isEmpty()) {
        lbwrong.setText("Ingrese sus credenciales.");
    } else {
        lbwrong.setText("Usuario o Contraseña Incorrecto!");
    }
}
```

Fuente: Elaboración Propia, 2022

Método GenerarGrafo: Este método se encarga de generar un archivo dot y luego convertirlo a imagen para la visualizarlo en la interfaz gráfica.

```
public void GenerarGrafo(String Capa){
    FileWriter fichero = null;
    PrintWriter pw = null;
    /*Crea un archivo con extesión .dot con el texto de la variable graph*/
    try{
        fichero = new FileWriter( fileName: "Capas.dot");
        pw = new PrintWriter(fichero);
        pw.write(graph);
        pw.close();
        fichero.close();
    }catch(Exception e){
        System.out.println("Error en generar dot de la capa");
    }finally {
        if(pw!=null){
            pw.close();
        }
    }
    try{

        ProcessBuilder proceso;
        String dir = Paths.get( first: "")
                .toAbsolutePath()
                .toString();
        proceso = new ProcessBuilder( ...command: "dot", "-Tpng", "-o", dir+"/"+"Capa"+Capa+".png","Capas.dot");
        proceso.redirectErrorStream(true);
        proceso.start();
    }catch (Exception e){
```

Fuente: Elaboración Propia, 2022

Método graficar: Este método es el encargado de guardar en una variable el contenido que tendrá el archivo dot

```
public void graficar(String Capa){
    int x = 0;
    int y = 0;
    graph = "graph G {\n" +
            "graph [dpi = 300.00 ];\n"+
            "node [shape=plaintext];\n" +
            "label=\"Capa "+Capa+"\";\n" +
            "some_node [\n" +
            "label=<\n" +
            "<table border=\"0\" cellborder=\"0\" cellspacing=\"0\" width=\"100%\" height=\"100%\">\n";

    while (y != (MaxCol+1)){
        graph += "<tr>\n";
        while (x != (MaxFila+1)){
            Nodo pixel = buscarpixel(x, y);
            if(pixel != null){
                graph += "  <td bgcolor=\""+pixel.color+"\" width=\"1\" height=\"1\"></td>\n";
            }else{
                graph += "  <td bgcolor=\"white\" width=\"1\" height=\"1\"></td>\n";
            }
            x++;
        }
        x =0;
        graph += "</tr>\n";
        y++;
    }

    graph += "</table>>\n" +
```

Fuente: Elaboración Propia, 2022

Métodos to: Estos métodos son los encargados de cambiar de ventana en ventana dependiendo de la clase en que se encuentren.

```
public void toAlbum(ActionEvent actionEvent) {
}

public void toImg() throws IOException {...}

public void toCapa() throws IOException {...}

public void toLogin() throws IOException {...}
```

Fuente: Elaboración Propia, 2022

REQUISITOS Y LIBRERIAS USADAS

JSON SIMPLE: Esta librería de Java es para procesamiento JSON, lectura y escritura de datos JSON y cumplimiento total con la especificación JSON

```
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
```

Fuente: Elaboración Propia, 2022

JavaFX: es una familia de productos y tecnologías de Oracle Corporation, para la creación de Rich Internet Applications, esto es, aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas.

```
import javafx.event.ActionEvent;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.ProgressBar;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
```

Fuente: Elaboración Propia, 2022

Este programa fue realizado con sistema operativo Linux pero pensado para ejecutarse en Windows, el JAR necesita pocos requisitos; la unica obligación es tener javas mayores a la 8.

Para seguir desarrollando este programa se debe utilizar Intellij con java 17 y tener instalado javafx.