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

Raudy David Cabrera Contreras
201901973

Manual Técnico

Clases Utilizadas:

- Cliente: Esta clase se utiliza como un objeto y como un nodo, ya que al usarse en todas las estructuras, el nodo de cada una es un tipo de nodo cliente.
- Lugar: Esta clase se utiliza como un objeto para las estructuras.
- Mensajero: Esta clase se utiliza como objeto para la tabla hash.
- Ruta: Esta clase se utiliza como objeto para la lista adyacente.
- Main: Esta clase se utiliza como centro de la aplicación.
- Árbol B: Clase utilizada para el control de los clientes.
- ListaAdyacencia: Clase utilizada para la lista adyacente para las rutas.
- Routes: Clase utilizada para el grafo de las rutas.
- Menu: Clase Frame utilizada para el menú de los clientes.
- Registrar: Clase Frame utilizada para que un usuario pueda registrarse
- Mod: Clase Frame utilizada para poder modificar datos de un cliente.
- Admin: Clase Frame utilizada para que el administrador pueda manejar los clientes.

Métodos

Clase **Cliente**

Los atributos de esta clase son el nombre, id del cliente, cantidades de imágenes a color y blanco/negro, una secuencia de ellos, pasos de cada cliente, ventanilla a la que ira y el estado que esta. Los atributos de sig y ant, son utilizados como apuntadores para las estructuras. El atributo pila es la pila que tendrá cada cliente. En el constructor se inicializan los atributos de cada uno y se crean set's y get's para cada atributo.

```
public class Cliente {
    private long dpi;
    private long telefono;
    private long idMuni;
    private String nombre;
    private String usuario;
    private String correo;
    private String contraseña;
    private String direccion;


    public Cliente(long dpi, String nombre, String usuario, String correo, String contraseña, long telefono, String direccion, long i
        this.dpi = dpi;
        this.telefono = telefono;
        this.nombre = nombre;
        this.usuario = usuario;
        this.correo = correo;
        this.contraseña = contraseña;
        this.direccion = direccion;
```

Clase Lugar

Los atributos de esta clase son el id del lugar, el departamento, el nombre y si tiene sucursal. Cuenta con sus set's y get's, en el constructor se inicializan.

```
public class Lugar {
    private int id;
    private String departamento;
    private String nombre;
    private String sucursal;

    public Lugar(int id, String departamento, String nombre, String sucursal) {
        this.id = id;
        this.departamento = departamento;
        this.nombre = nombre;
        this.sucursal = sucursal;
    }

    ...
```

Clase Mensajero

Esta clase posee los atributos de nombres, apellidos, licencia, genero, teléfono, dirección, dpi, y un nodo siguiente de tipo mensajero. Cada uno posee sus set's y get's.

```
public class Mensajero {

    private String nombres,apellidos,licencia,genero,telefono,direccion;
    private long dpi;
    private Mensajero siguiente;

    public Mensajero(long dpi, String nombres, String apellidos, String licencia, String genero, String telefono, String direccion) {
        this.nombres = nombres;
        this.apellidos = apellidos;
        this.licencia = licencia;
        this.genero = genero;
        this.telefono = telefono;
        this.direccion = direccion;
        this.dpi = dpi;
        //this.siguiente = siguiente;
    }

    /**
     * @return the nombres
```

Clase Ruta

Posee 3 atributos, uno de inicio, otro de fin y uno de peso. Posee cada uno sus set's y get's.

```java
public class Ruta {
    private int inicio;
    private int fin;
    private int peso;

    public Ruta(int inicio, int fin, int peso) {
        this.inicio = inicio;
        this.fin = fin;
        this.peso = peso;
    }

    /**
     * @return the inicio
     */
    public int getInicio() {
```

Clase Inicio

Esta clase es el inicio de la aplicación, donde se escribe el usuario y la contraseña. Cuenta con métodos como:

Entrar: Que sirve para validar los datos para iniciar sesión.

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

    if(usuario.equals("1999")){
        if(contra.equals("2022")){
            Admin admin = new Admin();
            admin.setVisible(true);
            dispose();
        }
    }else{
        long user = Long.parseLong(usuario);
        Cliente cl = Main.clientes.buscar(user);
        //Cliente cl = Main.clientes.buscar22(usuario);

            if(cl!=null){
```

Registrar: Método utilizado para abrir la ventana para registrarse.

```java
private void btnRegistrarActionPerformed(java.awt.event.ActionEvent evt) {
    Registrar R = new Registrar();
    R.setVisible(true);
    dispose();
}
```

Clase Menu

Es la clase de menú de los usuarios al ingresar en la aplicación. Cuenta con métodos como:

Carga de Lugares: Metodo donde se usa la librería de jsonsimple para el análisis de los json y convertirlos en objetos para las estructuras.

```
public void cargar_lugares(File archivo) {
    try {
        JSONParser parser = new JSONParser();
        FileReader leido = new FileReader(archivo);
        Object ob = parser.parse(leido);
        JSONObject js = (JSONObject) ob;
        String key = "";
        JSONObject job;
        key = "" + js.get("Lugares");
        Object obj = parser.parse(key);
        JSONArray array = (JSONArray) obj;
        //System.out.println("Tamaño: " + array.size());
        for (int i = 0; i < array.size(); i++) {
            System.out.println("-----**LUGAR**------");
            job = (JSONObject) array.get(i);
            String id = job.get("id").toString();
            String name = job.get("nombre").toString();
            String depa = job.get("departamento").toString();
            String sucursal = job.get("sn_sucursal").toString();
            System.out.println("Departamento: " + depa + "  Nombre: "
```

Carga de Rutas: Metodo donde se usa la librería de jsonsimple para el análisis de los json y convertirlos en objetos para las estructuras.

```
public void cargar_mensajeros(File archivo) throws FileNotFoundException, ParseException
    Scanner le = new Scanner(archivo);
    String linea = "";
    while (le.hasNextLine()) {
        linea += le.nextLine() + "\n";
    }
    JSONParser parser = new JSONParser();
    Object ob = parser.parse(linea);
    JSONArray array = (JSONArray) ob;
    JSONObject job;
    String dpi = "", nombres = "", apellidos = "", licencia = "", genero = "", direccion
```

Carga de Mensajeros: Metodo donde se usa la librería de jsonsimple para el análisis de los json y convertirlos en objetos para las estructuras.

```
public void cargar_rutas(File archivo) {
    try {
        JSONParser parser = new JSONParser();
        FileReader read = new FileReader(archivo);
        Object ob = parser.parse(read);
        JSONObject js = (JSONObject) ob;
        String key = "";
        JSONObject job;
        key = "" + js.get("Grafo");
        Object obj = parser.parse(key);
        JSONArray array = (JSONArray) obj;
        System.out.println("Tamaño: " + array.size());

        Routes rut = new Routes();
        ListaAdyacencia la = new ListaAdyacencia(array.size()

        for (int i = 0; i < array.size(); i++) {
```

Modificar Datos: Método que abre la ventana para modificar al usuario.

```
private void btnModActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    ModCliente actual = new ModCliente();
    actual.llenarlo(Main.actual.getDpi(), Main.actual.getNombre(), Main.a
    actual.setVisible(true);
}
```

Clase Crear Cliente

Esta clase se utiliza en Frame para la ventana con el cual se creara el cliente de forma manual, cuenta con el método de crear:

```
private void btnCrearActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(txtdpi.getText().equals("") || txtNombre.getText().equals("") || txtdpi.getText().equals("")){
        JOptionPane.showMessageDialog(null,"Complete el Formulario.","Error",JOptionPane.ERROR_MESSAGE);
    }else{
        try {
            long dpi = Long.parseLong(txtdpi.getText());
            String nombre = txtNombre.getText();
            String pass = txtPass.getText();
            String correo = txtCorreo.getText();
            long tel = Long.parseLong(txtCel.getText());
            String direccion = txtDireccion.getText();
            long muni = Long.parseLong(txtMuni.getText());
            String user = txtUser.getText();
            Cliente nuevo = new Cliente(dpi,nombre,user,correo,pass,tel,direccion,muni);
```

Clase Registrar

Esta clase tipo Frame se utiliza para que un usuario pueda crear uno, cuenta con el método para crear y regresar al menú.

```
private void btnCrearActionPerformed(java.awt.event.ActionEvent evt)
    // TODO add your handling code here:
    if(txtNombre.getText().equals("") || txtDpi.getText().equals("")
        JOptionPane.showMessageDialog(null,"Ingrese Todos Los Campos"
    }else{
        try {
            long dpi = Long.parseLong(txtDpi.getText());
            String nombre = txtNombre.getText();
            String pass = txtPass.getText();
            String correo = txtCorreo.getText();
            long tel = Long.parseLong(txtCel.getText());
            String direccion = txtDireccion.getText();
            long muni = Long.parseLong(txtMuni.getText());
            String user = txtUser.getText();
            Cliente nuevo = new Cliente(dpi,nombre,user,correo,pass,t
```

Clase Modificar Cliente

Esta clase se utilizada tipo Frame para la visualización de modificar los datos de un cliente, cuenta con el método para hacer los cambios:

```
private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(txtdpi.getText() != null && txtNombre.getText() != null && txtPass.getText() != null &&
        try {
            String dd = txtdpi.getText();
            long dpi = Long.parseLong(dd);
            String nombre = txtNombre.getText();
            String contra = txtPass.getText();
            String correo = txtCorreo.getText();
            long tel = Long.parseLong(txtCel.getText());
            String direccion = txtDireccion.getText();
            long muni = Long.parseLong(txtMuni.getText());
            String user = txtUser.getText();
```

Clase Árbol B

Esta clase se utiliza para llevar el control de los clientes, cuenta con los siguientes métodos:

Insertar: En este se inserta el cliente en el árbol, se recorre para ver si ya existe o no.

```
public void insertar(Cliente cl) {
    Cliente n = brecursivo(cl.getDpi(), raiz);
    if (n != null) {
        JOptionPane.showMessageDialog(null, "El Cliente con DPI: " + n.getDpi(
    } else {
        NodoB aur = raiz;
        insertarLL(raiz, cl);
        if (lleno(raiz)) {
            NodoB aus = new NodoB();
            aus.hoja = false;
            raiz = aus;
            raiz.n0 = aur;
            split(raiz, 0, aur);
        }
    }
```

Buscar: Este método se utiliza para ubicar un cliente, si ya existe lo devuelve para validarlo.

```
public Cliente brecursivo(long dpi, NodoB raiz) {
    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;
```

Actualizar: este método funciona para cuando se hace la modificación de un cliente, se recorre de forma recursiva para hacer más rápido el proceso.

```
public void actualizarRecur(Cliente cl, NodoB raiz) {
    NodoB aux = raiz;
    if (aux.info1 != null && aux.info1.getDpi() == cl.getDpi()) {
        aux.info1 = cl;
    } else if (aux.info2 != null && aux.info2.getDpi() == cl.getDpi()) {
        aux.info2 = cl;
    } else if (aux.info3 != null && aux.info3.getDpi() == cl.getDpi()) {
        aux.info3 = cl;
    } else if (aux.info4 != null && aux.info4.getDpi() == cl.getDpi()) {
        aux.info4 = cl;
    } else {
        if (hijos(aux)) {
            if (aux.info1 == null || (aux.n0 != null && aux.info1.getDpi() >
```

Clase Lista Adyecencia

Esta clase cuenta con los atributos de vértices y max. Que se pide al llamarla al momento de crearla, cuenta con los métodos de:

Insertar: en este método se pasa como parámetro de donde viene hacia donde va y su distancia.

```
public void insertaArista(int v1, int v2, int dist)
        throws ArrayIndexOutOfBoundsException, Unsuppo
    if (v1 >= maxVer || v2 >= maxVer) {
        throw new ArrayIndexOutOfBoundsException(
                "Vertices inválidos, fuera de rango" +
    } else if (aristas == maxAri) {
        throw new UnsupportedOperationException("No se
    } else {
        matrix[v1][v2] = dist;
    }
}
```

Eliminar: esta se utiliza para borrar una conexión de inicio a fin.

```
public void eliminaArista(int v1, int v2) {
    if (v1 >= maxVer || v2 >= maxVer) {
        throw new ArrayIndexOutOfBoundsException(
                "Vertices inválidos, fuera de rango"
    } else if (matrix[v1][v2] == 0) {
        System.err.println("La arista NO existe");
    } else {
        matrix[v1][v2] = 0;
    }
}
```

Existe: utilizada para saber si existe esa conexión ya.

```
public void eliminaArista(int v1, int v2) {
    if (v1 >= maxVer || v2 >= maxVer) {
        throw new ArrayIndexOutOfBoundsException(
                "Vertices inválidos, fuera de rang
    } else if (matrix[v1][v2] == 0) {
        System.err.println("La arista NO existe");
    } else {
        matrix[v1][v2] = 0;
    }
}
```

Clase Lista

Esta clase se utilizo para hacer la lista de la tabla hash. Cuenta con un nodo cabecera tipo mensajero. Cuenta con los siguientes métodos.

Add: es el método para insertar mensajeros.

```
public void add(Men emp) {
    if (head == null) {
        head = emp;
        return;
    }
    Men temp = head;
    while (temp.siguiente != nu
        // hacia atrás
        temp = temp.siguiente;
    }
    temp.siguiente = emp;
}
```

Buscar: este busca por el dpi el mensajero y lo devuelve.

```
if (head == null) {
    return null;
}
Men temp = head;
while (true) {
    if (temp.dpi.equals(id)) {
        break;
    }
    if (temp.siguiente == null) {
        temp = null;
        break;
    }
    // hacia atrás
    temp = temp.siguiente;
}
```

Clase Tabla

Esta clase se utilizo para la tabla hash, donde el constructor inicia con la Lista para hash.

```
public class Tabla {
    private Lista[] empLinkedListArray;
    private int size;

    public Tabla(int size) {
        this.empLinkedListArray = new Lista[size];
        this.size = size;
        for (int i = 0; i < size; i++) {
            empLinkedListArray[i] = new Lista();
        }
    }
```

Add: este método se utiliza para agregar elementos a la tabla.

```
public void add(Men emp) {
    int a = emp.dpi.intValue();
    int empLinkedListNO = hashFun(a);
    empLinkedListArray[empLinkedListNO].add(emp)
}
```

Buscar: para encontrar información en la tabla.

```
public void buscar(int id) {
    int is = hashFun(id);
    Men emp = empLinkedListArray[is].finc
    if (emp != null) {
        System.out.println("id valor corı
    }
}
```