UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS

ESTRUCTURAS DE DATOS


Nombre: Joseph Jeferson Marroquín Monroy
Carné: 202010316

# Manual Técnico

**Lugar, fecha y responsables de la elaboración**

El Proyecto 1 Fase 1 fue desarrollado entre el 1 y el 20 de febrero del 2022, elaborado por Joseph Jeferson Marroquin Monroy.

**Objetivos**

El presente manual tiene como finalidad describir la funcionalidad del algoritmo aplicado para simular la estancia de un cliente dentro de una empresa.

**Configuración del sistema**

El proyecto 2 está desarrollado en una Laptop Lenovo Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz  2.40 GHz  con 8GB de RAM, Sistema operativo de 64 bits, procesador x64.

**Menú Inicial**

```
Scanner sn = new Scanner(System.in);
boolean salir = false;
int opcion;

while (!salir) {

    System.out.println("-----------------------------------------");
    System.out.println("| 1. Parámetros iniciales                |");
    System.out.println("| 2. Ejecutar paso                       |");
    System.out.println("| 3. Estado en memoria de las estructuras |");
    System.out.println("| 4. Reportes                            |");
    System.out.println("| 5. Acerca de                           |");
    System.out.println("| 6. Salir                               |");
    System.out.println("-----------------------------------------");

    try {

        System.out.println("Introduce una opcion");
        opcion = sn.nextInt();

        switch (opcion) {
            case 1:
                parametrosIniciales();
                break;
```

Usamos Scanner para leer la cadena de texto, por medio de un while creamos nuestros menús y por el switch asignamos a donde nos va a dirigir la opción que se elija.

```
String contenidoJSON = "";
try (BufferedReader br = new BufferedReader(new FileReader(path))) {
    String linea;
    while ((linea = br.readLine()) != null) {
        contenidoJSON += linea;
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

Leemos nuestro archivo JSON guardándolo en una cadena de texto.

```
//EXTRACCION DEL JSON EN EL STRING
for (int i = 1; i <= ntotalClientes; i++) {

    //Insertar clientes en nuestra Cola Recepcion
    String nCliente = "Cliente" + i;
    JSONObject objetoJson = new JSONObject(contenidoJSON);
    JSONObject contenidoCliente = objetoJson.getJSONObject(nCliente);
    int id_cliente = contenidoCliente.getInt("id_cliente");
    String nombre_cliente = contenidoCliente.getString("nombre_cliente");
    int img_color = contenidoCliente.getInt("img_color");
    int img_bw = contenidoCliente.getInt("img_bw");

    ClientesEnCola cl = new ClientesEnCola(nCliente, id_cliente, nombre_cliente, img_color, img_bw);
    cola_recepcionVerdad.Insertar(cl);

}
```

Extraemos los datos contenidos en el JSON y lo guardamos en nuestra estructura que en este caso seria una lista tipo cola.

```
public static void cargarVentanillas(int nVentanas) {
    for (int i = 1; i <= nVentanas; i++) {
        Ventanillas vl = new Ventanillas(i);
        lista_ventanillas.InsertarVentanilla(vl);

        ImagenPorVentana imgl = new ImagenPorVentana(i);
        lista_img_pila.Insertar_ImagenPorVentana(imgl);
    }
}
```

Al momento que ingresamos la cantidad de ventanillas disponibles generamos dos estructuras: una lista para las ventanillas y una pila para cada ventana que almacenara las imágenes dadas por el cliente.

```
ublic static void EjecutarPaso() {

    //-------------------------------------------------------------------------------------
    cola_recepcionVerdad.MostrarEncabezadoCliente(lista_ventanillas); //Saber que cliente estoy atendiendo
    String encabezadoCliente = ColaRecepcionVerdad.encabezadoCliente; //Saber que cliente estoy atendiendo
    int idDelCliente = ColaRecepcionVerdad.idDelCliente; //Saber que cliente estoy atendiendo
    lista_ventanillas.ingresarClienteVentana(encabezadoCliente, idDelCliente, cola_recepcion, cola_recepcionVerd
    //-------------------------------------------------------------------------------------
```

Sacamos al cliente de la cola de recepción y lo mandamos a una ventanilla disponible donde entregara sus imágenes.

```
ublic static void GenerarClientesColaRecepcion() {
    //Genero clientes aleatorios (entre 0 y 3)
    int CantidadClientes = 0;
    CantidadClientes = (int) (Math.random() * 4);

    //Genero cantidad de imagenes por cliente (entre 0 y 4)
    int ImgColor = 0;
    ImgColor = (int) (Math.random() * 3);

    int ImgBw = 0;
    ImgBw = (int) (Math.random() * 3);

    String nombre[] = {"Joseph", "Luis", "Anthony", "Maria", "Jeferson", "Monica", "Sofia", "Alejandra", "Steve

    int NombreAlz = 0;
    NombreAlz = (int) (Math.random() * nombre.length);

    String encabezado;


    for (int i = 0; i < CantidadClientes; i++) {

        int id = 0;
        id = (int) (Math.random() * (10000 - 1000 + 1) + 1000);
        int nuevo_id = cola_recepcionVerdad.VerificacionId(id);

        while (id != nuevo_id) {
            nuevo_id = cola_recepcionVerdad.VerificacionId(id);
        }

        encabezado = "Cliente" + id;


        ClientesEnCola clienteCola = new ClientesEnCola(encabezado, nuevo_id, nombre[NombreAlz], ImgColor, ImgB
        cola_recepcionVerdad.Insertar(clienteCola);
    }
```

Generamos clientes aleatorios al momento de ejecutar cada paso.

```
public class ClientesEnCola {
    String encabezado;
    int id_cliente;
    String nombre_cliente;
    int img_color;
    int img_bw;
    String atendiendo;

    public ClientesEnCola(String _encabezado,int _id_cliente, String _nombre_cliente, int _img_color, int _
        this.encabezado=_encabezado;
        this.id_cliente=_id_cliente;
        this.nombre_cliente=_nombre_cliente;
        this.img_color=_img_color;
        this.img_bw=_img_bw;
        this.atendiendo="no";
    }
}
```

Generamos constructores donde almacenaremos la información.

```
private Nodo inicioCola, finalCola;
String Cola = "";

public class Nodo {

    public ClientesEnCola clientesEnCola;
    public Nodo siguiente = null;

    public Nodo(ClientesEnCola _ClientesEnCola) {
        this.clientesEnCola = _ClientesEnCola;
    }

}
```

Creamos una clase nodo para nuestras diferentes estructuras.

```
//Metodo para insertar a la cola
public void Insertar(ClientesEnCola _clientesEnCola) {
    Nodo nuevo_nodo = new Nodo(_clientesEnCola);
    nuevo_nodo.clientesEnCola = _clientesEnCola;
    nuevo_nodo.siguiente = null;

    if (ColaVacia()) {
        inicioCola = nuevo_nodo;
        finalCola = nuevo_nodo;
    } else {
        finalCola.siguiente = nuevo_nodo;
        finalCola = nuevo_nodo;
    }

}
```

Creamos métodos para ingresar datos dentro de las estructuras y así poder guardar los datos.

```
//Metodo para graficar en graphviz
public void generarDot() throws IOException{
    String resultado="digraph G{\nlabel=\""+"Lista de ventanillas"+"\";\nnode [shape=box];\n";
    Nodo aux = cabeza;
    String conexiones="";
    String nodos="";
    while(aux != null){
        nodos+="N"+aux.hashCode()+"[label=\"nodo "+" Ventanilla "+aux.ventanilla.nVentanilla+"\"];\n";
        if(aux.next != null){
            conexiones+="N"+aux.hashCode()+ " -> "+"N"+aux.next.hashCode()+";\n";
        }
        aux = aux.next;
    }
    resultado+= "//Agregando nodods\n";
    resultado+=nodos+"\n";
    resultado+= "//Agregando conexiones\n";
    resultado+="{rank= same;\n"+conexiones+"\n";

    resultado+="}\n}";

    String path = "Estructuras\\ListaVentanillas.txt";
    Files.write(Paths.get(path), resultado.getBytes());

}
```

Generamos los archivos .dot para graficar las estructuras con graphviz

```
public void generarJPG(){
    try{
        String dotPath = "C:\\Program Files\\Graphviz\\bin\\dot.exe";

        String fileInputPath ="Estructuras\\ListaVentanillas.txt";
        String fileOutputPath = "Estructuras\\ListaVentanillas.jpg";

        String tParam = "-Tjpg";
        String tOParam = "-o";

        String[] cmd = new String[5];

        cmd[0] = dotPath;
        cmd[1] = tParam;
        cmd[2] = fileInputPath;
        cmd[3] = tOParam;
        cmd[4] = fileOutputPath;

        Runtime rt = Runtime.getRuntime();
        rt.exec(cmd);

    }catch (Exception ex) {
        ex.printStackTrace();
    } finally {}
}
```

Generamos un jpg donde se muestra el grafo generado por el .dot

```
public void ingresarImagenApila(ColaRecepcion cola_recepcion, ListaImgPila lista_img_pila, ColaImpresion
    Nodo aux=cabeza;
    while(aux!=null){

        if(aux.ventanilla.estado=="Ocupado"){

            //Mando a traer datos del cliente en recepcion
            int imagenesAcolor=cola_recepcion.CantidadImgColorRecepcion(aux.ventanilla.id_cliente);
            int imagenesAbw=cola_recepcion.CantidadImgBwRecepcion(aux.ventanilla.id_cliente);
            String encabezado=cola_recepcion.EncabezadoParaImg(aux.ventanilla.id_cliente);
            String nombreCliente=cola_recepcion.NombreParaImg(aux.ventanilla.id_cliente);
            //

            if(imagenesAcolor!=0){
                System.out.println("La ventanilla "+aux.ventanilla.nVentanilla+" recibe una imagen a colo
                cola_recepcion.QuitarImagenColor(aux.ventanilla.id_cliente);

                //agregar imagen a la pila
                ImagenPorVentana imv=new ImagenPorVentana(0);
                imv=lista_img_pila.BuscarVentana(aux.ventanilla.nVentanilla);
                Imagenes cc=new Imagenes(aux.ventanilla.id_cliente,"color");
                imv.pilaImagen.InsertarPilaImg(cc);
            }
            else if(imagenesAcolor==0){
                if(imagenesAbw!=0){
                    System.out.println("La ventanilla "+aux.ventanilla.nVentanilla+" recibe una imagen en
                    cola_recepcion.QuitarImagenBw(aux.ventanilla.id_cliente);

                    //agregar imagen a la pila
                    ImagenPorVentana imv2=new ImagenPorVentana(0);
                    imv2=lista_img_pila.BuscarVentana(aux.ventanilla.nVentanilla);
                    Imagenes cc2=new Imagenes(aux.ventanilla.id_cliente,"blanco y negro");
                    imv2.pilaImagen.InsertarPilaImg(cc2);
                }
```

Creamos un algoritmo que le quitara una imagen al cliente cada paso que esta en la ventanilla.

```
//INGRESO DEL CLIENTE A LAS VENTANILLAS
public void ingresarClienteVentana(String encabezadoCliente, int idCliente, ColaRecepcion colaRecepcion,
    Nodo aux=cabeza;
    while(aux!=null){

        if(aux.ventanilla.estado=="Disponible" && ColaRecepcionVerdad.encabezadoCliente!=""){
            aux.ventanilla.estado="Ocupado"; //Se cambia el estado de la ventanilla a ocupado
            aux.ventanilla.id_cliente=idCliente; //La ventanilla guarda que cliente esta atendiendo
            System.out.println("El "+encabezadoCliente+" ingresa a la ventanilla "+aux.ventanilla.nVenta

            ClientesEnCola clientCola=new ClientesEnCola("",0,"",0,0);
            clientCola=colaRecepcionVerdad.mandarAlClinete(idCliente);

            Clientes client=new Clientes(clientCola.encabezado,clientCola.id_cliente,clientCola.nombre_c
            colaRecepcion.Insertar(client);

            colaRecepcionVerdad.Extraer();

            break;
        }

        aux=aux.next;
    }
}
```

Verificamos que la ventanilla este disponible para que un cliente nuevo pueda ingresar.