Manual Técnico

Proyecto Fase 1

Kevin Steve Martinez Lemus
[EDD] Universidad De San Carlos De Guatemala

Índice

I. Introducción

El presente documento describe los aspectos técnicos informáticos del sistema de información. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

Objetivo

Instruir el uso adecuado del Sistema de Información, para el acceso oportuno y adecuado en la modificación de este, mostrando la descripción de los archivos relevantes del sistema los cuales nos orienten en la configuración y soporte de este.

Requerimientos

- Tener Java (versión 8 o superiores) instalado.
- Se recomienda tener un editor de código (NetBeans 8.2 u otro) para ver y editar el código de este programa.

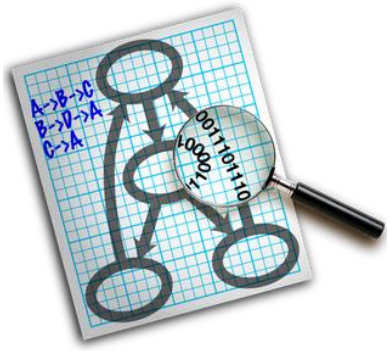
II. Descripción Del Sistema

1. Descripción del Contenido del Sistema

Este es un sistema en el cual el usuario interactúa directamente con un menú en consola, este programa permite realizar una simulación, teniendo como parámetros iniciales un archivo JSON con clientes y la creación de ventanillas virtuales, realizando pasos para la ejecución de la simulación, así como generar 4 diferentes reportes y graficar el estado en memoria de las estructuras realizadas.

2. Lenguajes Utilizados

Para la creación de este sistema todas las funcionalidades se realizaron en lenguaje de JAVA y también se utilizó Graphviz para la creación de reportes. Utilizando un enfoque al paradigma de programación POO (Programación Orientada a Objetos). Creando las diferentes estructuras de datos para almacenar la información.



Todo realizado en el IDE NetBeans 8.2:

3. Menú en consola

Para la realización del menú en consola, se utilizó un ciclo while el cual termina al leer la opción de salida:

```
while (!opcion.equals("6")) {
    System.out.println("\n------------ MENÚ PRINCIPAL ------------");
    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("----------------------------------------");
    System.out.println("Elija Una Opción\n");
```

Para la lectura de la opción se utilizó un scanner utilizado de la siguiente manera:

```
import java.util.Scanner;
```

```
// Lector de opciones
Scanner leer = new Scanner(System.in);
```

```
opcion = leer.nextLine();
```

Y para la verificación de la opción leída se utilizó la estructura de switch:

```
opcion = leer.nextLine();

switch(opcion){
    case "1":
```

4. Lectura de Archivo

Para la lectura de archivo JSON se utilizó la librería de gson-2.8.2:

La lectura del archivo se inicia pidiendo la dirección de la ubicación del archivo que se desea cargar, este archivo se abre y se lee con la ayuda de:

```
import java.io.File;
```

```
import java.io.BufferedReader;
```

Utilizados de la siguiente manera:

```
File doc = new File(ruta);
Scanner obj1 = new Scanner(doc);
String t = "";
int cont = 0;
boolean exist = true;

JsonParser parser = new JsonParser();

while (obj1.hasNextLine()){
    t += obj1.nextLine();
    t += "\n";
}
```

En donde el archivo queda guardado como un String dentro de la variable t, y JsonParser pertenece a la librería gson el cual nos ayuda a parsear el String a JSON, utilizado en el siguiente fragmento de código que se utilizó para generar los objetos de clientes:

```
JsonObject gsonObj = parser.parse(t).getAsJsonObject();

while(exist){
    cont += 1;
    try {
        JsonObject ej = gsonObj.get("Cliente"+cont).getAsJsonObject();

        int id = Integer.parseInt(ej.get("id_cliente").getAsString());
        int imgc = Integer.parseInt(ej.get("img_color").getAsString());
        int imgbw = Integer.parseInt(ej.get("img_bw").getAsString());
        String name = ej.get("nombre_cliente").getAsString();

        /*System.out.println("------------------------------");
        System.out.println("Cliente"+cont);
        System.out.println("ID: " + id);
        System.out.println("El Nombre es "+name);
        System.out.println("Imagenes a Color: " + imgc);
        System.out.println("Imagenes B&W: " +imgbw);*/

        Lista imgs = new Lista();

        for (int i = 0; i<=imgc-1 ; i++) {
            Imagen nImagenC = new Imagen(id,true);
            imgs.add(nImagenC);
        }

        for(int i=0;i<=imgbw-1;i++){
            Imagen nImgBW = new Imagen(id, false);
            imgs.add(nImgBW);
        }


        int cant = (imgc+imgbw);

        Cliente nuevoCliente = new Cliente(name,id,imgs,cant,imgc,imgbw);
        recepcion.add(nuevoCliente);

    } catch (Exception e) {
        exist = false;
    }
```

5. Objetos Utilizados

Las clases utilizadas para los objetos son las siguientes: de Cliente con los siguientes atributos:

```
public class Cliente {

    private String nombre;
    private int id;
    private Lista imgs;
    private int ventanilla;
    private int cant;
    private int bw;
    private int c;
    int pasos;

    public Cliente(String nombre, int id, Lista imgs, int cant, int c, int bw) {
        this.nombre = nombre;
        this.id = id;
        this.imgs = imgs;
        this.ventanilla = 0;
        this.cant = cant;
        this.pasos = 0;
        this.bw = bw;
        this.c = c;
    }
```

La clase Imagen, con id para saber de quién es la imagen y un atributo booleano para saber si es a color o a blanco y negro:

```
public class Imagen {
    private int id;
    private boolean tipo;

    public Imagen(int id, boolean tipo) {
        this.id = id;
        this.tipo = tipo;
    }
```

Y por último la clase Ventanilla:

```
public class Ventanilla {

    private int id;
    private Cliente cliente;
    private Pila imgs;
    boolean recibe;

    public Ventanilla(int id) {
        this.id = id;
        this.cliente = null;
        this.imgs = new Pila();
    }
```

6. Estructuras de datos utilizadas

Las estructuras que se utilizaron son la de Cola, Lista, Pila y Lista Circular; para cada una de ellas se creo una clase Nodo y diferentes métodos necesarios para cada uno de ellos. La estructura de Cola se utilizo para la cola de recepción, cola de impresora a color y cola de impresora a blanco y negro, sus métodos add, delete, tamaño ayudo a generar una estructura funcional:

```java
public class Cola {
    private Nodo cabecera;

    public class Nodo{
        Object info;
        Nodo next;

        public Nodo(Object info) {
            this.info = info;
            this.next = null;
        }
    }

    public void add(Object info){
            Nodo nodonuevo = new Nodo(info);
            if(cabecera == null){
                cabecera = nodonuevo;
            }
            else{
                Nodo aux = cabecera;
                while(aux.next != null){
                    aux=aux.next;
                }
                aux.next = nodonuevo;
            }
    }
```

La estructura de Lista se parece a la estructura de Cola, pero como esta se utilizó para la lista de ventanillas, lista de imágenes del cliente y la lista de clientes atendidos, esta posee diferentes métodos como buscar y vaciar:

```java
public class Lista {
    Nodo cabecera;

    public class Nodo{
        Object info;
        Nodo next;

        public Nodo(Object info) {
            this.info = info;
            this.next = null;
        }
    }

    public void add(Object info){
            Nodo nodonuevo = new Nodo(info);
            if(cabecera == null){
                cabecera = nodonuevo;
            }
            else{
                Nodo aux = cabecera;
                while(aux.next != null){
                    aux=aux.next;
                }
                aux.next = nodonuevo;
            }
    }
```

Con la estructura de Pila los métodos cambian ya que posee los métodos push y pop para el buen funcionamiento de la misma, utilizada para la pila de imágenes que posee la clase Ventanilla:

```
public class Pila {
    Nodo cabecera;

    public class Nodo{
        public Object info;
        public Nodo next = null;

        public Nodo(Object info){
            this.info = info;
        }
    }

    public void push(Object info){
        Nodo nuevonodo = new Nodo(info);
        nuevonodo.next = cabecera;
        cabecera = nuevonodo;
    }

    public Object pop(){
        Object aux = cabecera.info;
        cabecera = cabecera.next;
        return aux;
    }
```

Por último, pero no menos importante, la estructura de lista circular doblemente enlazada en donde cada nodo posee dos enlaces uno al siguiente y otro al anterior, y el ultimo elemento se conecta con el primero, utilizada para la lista de espera:

```
public class ListaCircular {
    private Nodo inicio;

    public class Nodo{
        Object info;
        Nodo next;
        Nodo anterior;

        public Nodo(Object info) {
            this.info = info;
            this.next = null;
            this.anterior = null;
        }
    }

    public void add(Object info){
        Nodo nuevo = new Nodo(info);

        if(inicio == null){
            inicio = nuevo;
            inicio.next = inicio;
            inicio.anterior = inicio;
        }else{
            Nodo aux = inicio;
            while(aux.next != inicio){
                aux = aux.next;
            }

            nuevo.anterior = aux;
            nuevo.next = inicio;
            aux.next = nuevo;
            inicio.anterior = nuevo;

        }
    }
```

7. Reportes

La generación de reportes se realizó con la herramienta de graphviz, el proceso es principalmente escribir la sintaxis deseada en un .txt, ya que cada estructura necesita un gráfico diferente, y ejecutándolo con el siguiente código:

```
try {

    String dotPath = "C:\\Program Files\\Graphviz\\bin\\dot.exe";
    String fileInputPath = System.getProperty("user.dir") + "\\"+title+".txt";
    String fileOutputPath = System.getProperty("user.dir") + "\\"+title+".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 );

    //System.out.println("Graficado");

} catch (Exception e) {
    e.printStackTrace();
}
```

Esto generará una imagen jpg con la estructura del reporte deseado.