UDrawing Paper

Manual Técnico

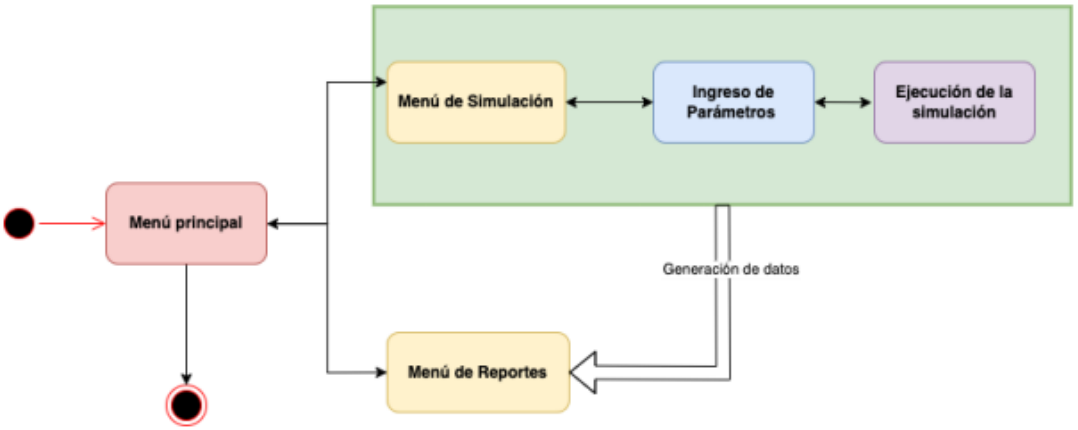
UDrawing Paper

Lenguaje Utilizado: Java 17
Sistema Operativo: Windows 7 en adelante

Resumen

UDrawing Paper es una aplicación diseñada para simular el comportamiento de colas que maneja una imprenta con el fin de observar los tiempos que conlleva cada parte del proceso. Además de eso, es capaz de crear un registro resumen de cada uno de los clientes que haya puesto un pie dentro de la empresa.

Flujo general del programa



Resumen de Métodos

GRAFICACIÖN (General para todas las estructuras):

Nombre	Descripción	Parámetros
declare	Lee los nodos y los declara usando su identificador único.	
connect	Declara las conexiones entre los nodos.	
getcodigo	Unifica las declaraciones y conexiones con el resto de la estructura del archivo	
escribir	Escribir el archivo de texto que va al cont-tener el código.	ruta (String) codigo (String)
dibujar	Ejecuta el comando en consola para invocar el proceso de graphviz. Recibe un *.txt .	entrada(String) salida(String)

CLIENTE:

Nombre	Descripción	Parámetros
getColor	Retorna el total de impresiones a color.	
getNegro	Retorna el total de impresiones B/N.	
getPasos	Retorna el total de pasos del cliente.	

COLA:

Nombre	Descripción	Parámetros
enqueue	Agrega un objeto a la cola..	Content(Object) Name (String)
dequeue	Regresa el objeto de la cabecera y la eliminar.	
isEmpty	Indica si la cabecera de la cola está vacía.	
delete	Elimina toda la cola.	

LISTAS:

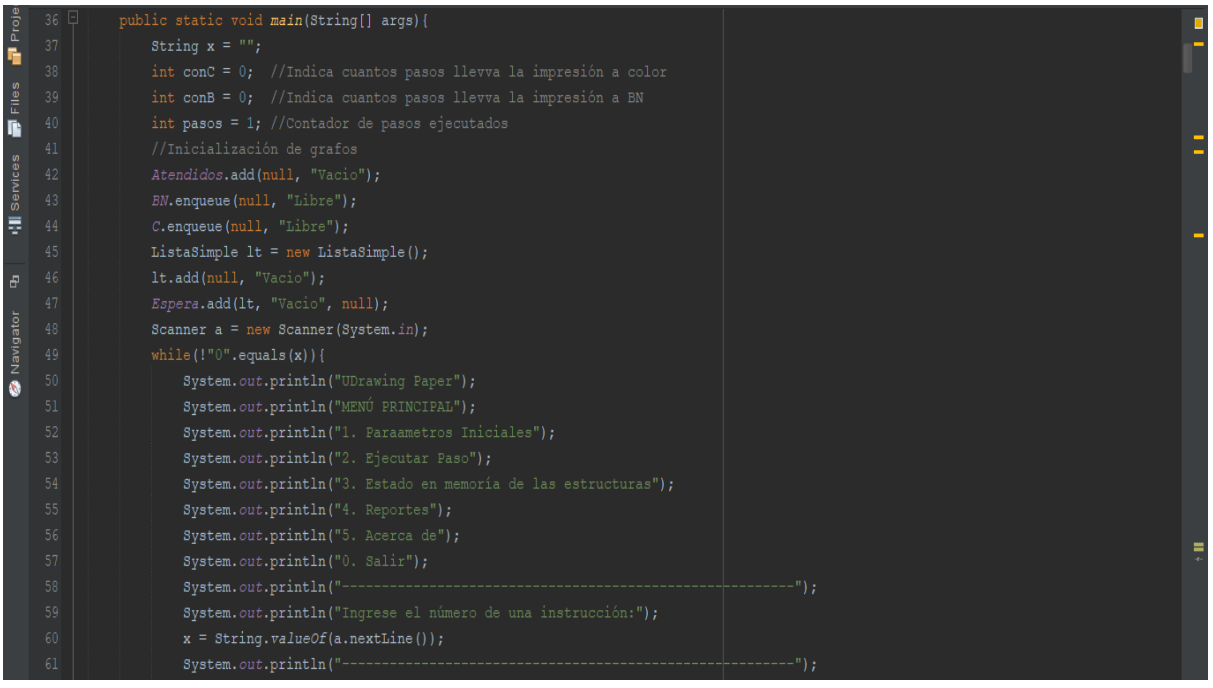
Nombre	Descripción	Parámetros
add	Agrega un objeto a la lista..	Content(Object) Name (String)
remove	Elimina el nodo indicado por el índice de entrada.	index (int)
isEmpty	Indica si la cabecera de la cola está vacía.	
deleteL	Elimina toda la cola.	
show	Imprime el contenido de la lista.	
find	Encuentra el nodo indicado por el índice de entrada.	index (int)

PILA:

Nombre	Descripción	Parámetros
push	Agrega un objeto a la pila..	Content(Object) Name (String)
pop	Regresa el objeto de la cabecera y la eliminar.	
isEmpty	Indica si la cabecera de la pila está vacía.	
deleteP	Elimina toda la pila.	

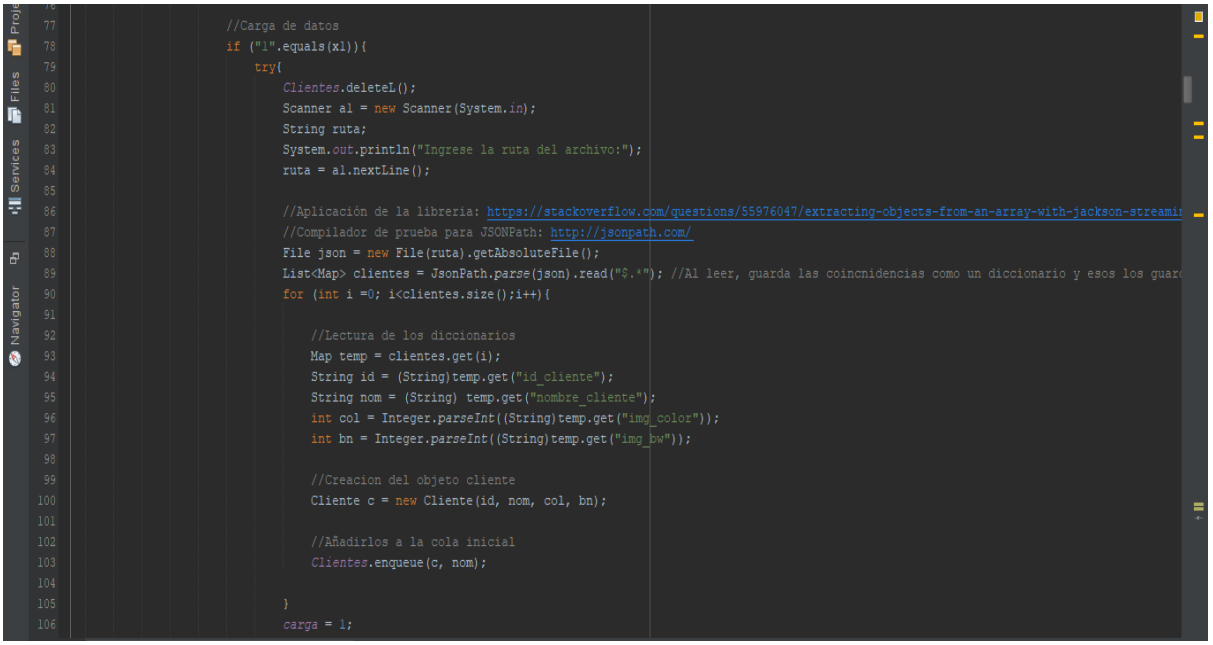
Explicación del flujo del programa:

El programa consiste, básicamente, en el llenado y vaciado de cada una de las estructuras que representan cada parte del proceso de la imprenta. Para manejo de los gráficos, estas nunca están vacías como tal, así que contienen un nodo con la etiqueta vacío que representa que no hay nada en la lista. Cada vez que se quiera trabajar con ella, se extrae este nodo y se agrega cuando se vacíe.



```
36    public static void main(String[] args){
37        String x = "";
38        int conC = 0;  //Indica cuantos pasos llevva la impresión a color
39        int conB = 0;  //Indica cuantos pasos llevva la impresión a BN
40        int pasos = 1; //Contador de pasos ejecutados
41        //Inicialización de grafos
42        Atendidos.add(null, "Vacio");
43        BN.enqueue(null, "Libre");
44        C.enqueue(null, "Libre");
45        ListaSimple lt = new ListaSimple();
46        lt.add(null, "Vacio");
47        Espera.add(lt, "Vacio", null);
48        Scanner a = new Scanner(System.in);
49        while(!"0".equals(x)){
50            System.out.println("UDrawing Paper");
51            System.out.println("MENÚ PRINCIPAL");
52            System.out.println("1. Paraametros Iniciales");
53            System.out.println("2. Ejecutar Paso");
54            System.out.println("3. Estado en memoría de las estructuras");
55            System.out.println("4. Reportes");
56            System.out.println("5. Acerca de");
57            System.out.println("0. Salir");
58            System.out.println("-------------------------------------------------------");
59            System.out.println("Ingrese el número de una instrucción:");
60            x = String.valueOf(a.nextLine());
61            System.out.println("-------------------------------------------------------");
```

El menú inicia mostrando el menú. Este es un ciclo while que continúa imprimiendo e ejecutando instrucciones hasta que se acceda la opción de salida. La primera opción del menú consiste en la carga de datos (clientes y ventanillas). El de clientes solicita que se cargue un archivo .json con los clientes iniciales de la estructura. Estos son almacenados en la cola de entrada.



```
//Carga de datos
if ("1".equals(xl)){
    try{
        Clientes.deleteL();
        Scanner al = new Scanner(System.in);
        String ruta;
        System.out.println("Ingrese la ruta del archivo:");
        ruta = al.nextLine();

        //Aplicación de la libreria: https://stackoverflow.com/questions/55976047/extracting-objects-from-an-array-with-jackson-streami
        //Compilador de prueba para JSONPath: http://jsonpath.com/
        File json = new File(ruta).getAbsoluteFile();
        List<Map> clientes = JsonPath.parse(json).read("$.*"); //Al leer, guarda las coincnidencias como un diccionario y esos los guar
        for (int i =0; i<clientes.size();i++){

            //Lectura de los diccionarios
            Map temp = clientes.get(i);
            String id = (String)temp.get("id_cliente");
            String nom = (String) temp.get("nombre_cliente");
            int col = Integer.parseInt((String)temp.get("img_color"));
            int bn = Integer.parseInt((String)temp.get("img_bw"));

            //Creacion del objeto cliente
            Cliente c = new Cliente(id, nom, col, bn);

            //Añadirlos a la cola inicial
            Clientes.enqueue(c, nom);

        }
        carga = 1;
```

El apartado de ventanilla no pide archivos. Para cargarlas, solo se necesita definir el número con el que se quiere trabajar.

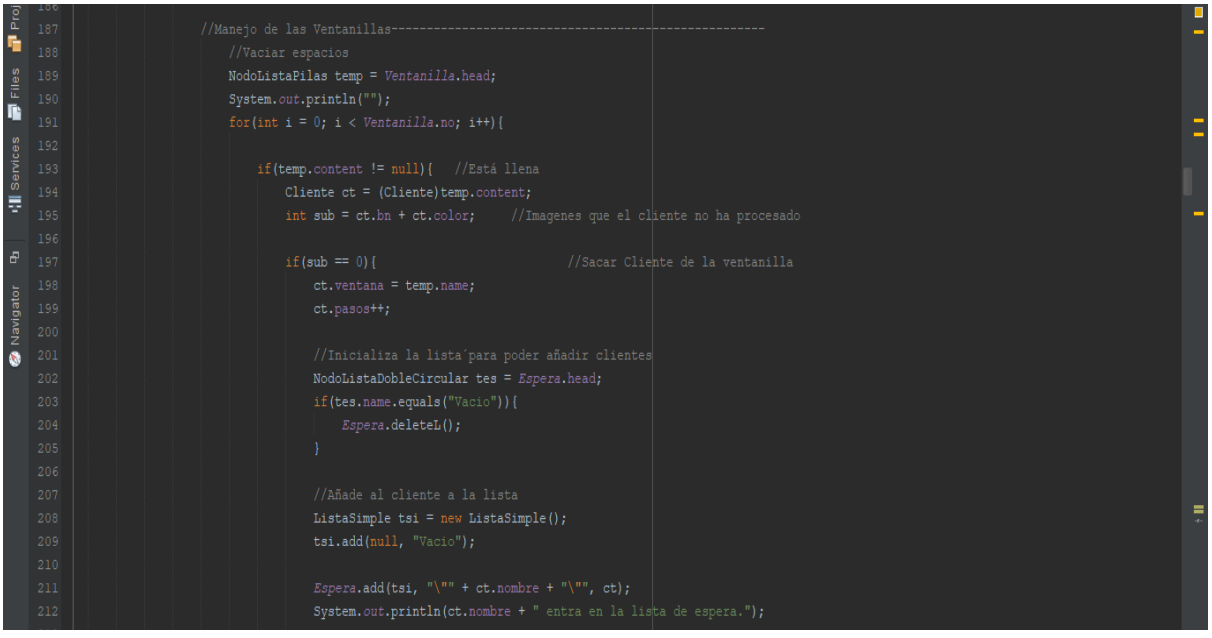
```
//Número de Ventanillas
if ("2".equals(xl)){
    System.out.println("Ingrese el número de ventanillas a manejar:");
    try{
        ventanas = a.nextInt();
        a.nextLine();
        Pila [] prueba = new Pila[ventanas];
        String [] ab = new String[ventanas];
        String n;
        for(int i = 0; i < ventanas; i++){
            prueba[i] = new Pila();
            prueba[i].push(null, "Libre");
            ab[i] = "Ventanilla_" + Integer.toString(i+1);
        }

        for(int i = 0; i < ventanas; i++){
            Ventanilla.add(prueba[i], ab[i]);
        }
        }catch(Exception e){

        }

        System.out.println("-----------------------------------------------------");
        System.out.println("");
        System.out.println("MENSAJE: Número de ventanillas actualizado.");
        System.out.println("");
        System.out.println("-----------------------------------------------------");
    }
}
```

La segunda opción se encarga del manejo de todas las estructuras (Tanto las entradas así como las salidas). Para un mejor análisis, se explicará por separado cada una de estas funciones.

Las colas solo manejan por su cuenta las entradas de los clientes. El único proceso a su cuenta es la de ingresar clientes..Se define aleatoriamente el número de clientes que entran así como sus atributos. Posterior a su creación, se añaden a la cola.

```
//Manejo de la Cola-------------------------------------------------------------
    //Inicializar cola
    NodoCola tcc = Clientes.head;
    if(tcc.name.equals("Vacio")){
        Clientes.deleteL();
    }

    //Añadir clientes nuevos
    Random r = new Random();
    int nuevos = r.nextInt(4);
    for(int i = 0; i < nuevos+1; i++){
        String nombre = Nombres[r.nextInt(Nombres.length)];
        String apellido = Apellidos[r.nextInt(Apellidos.length)];
        String NA = nombre + " " + apellido;
        String ID = UUID.randomUUID().toString();
        int bnc = r.nextInt(5);
        int cc = r.nextInt(5);
        Cliente c = new Cliente(ID, NA, bnc, cc);
        Clientes.enqueue(c, NA);
        System.out.println(NA + " entra a la empresa.");
    }
```

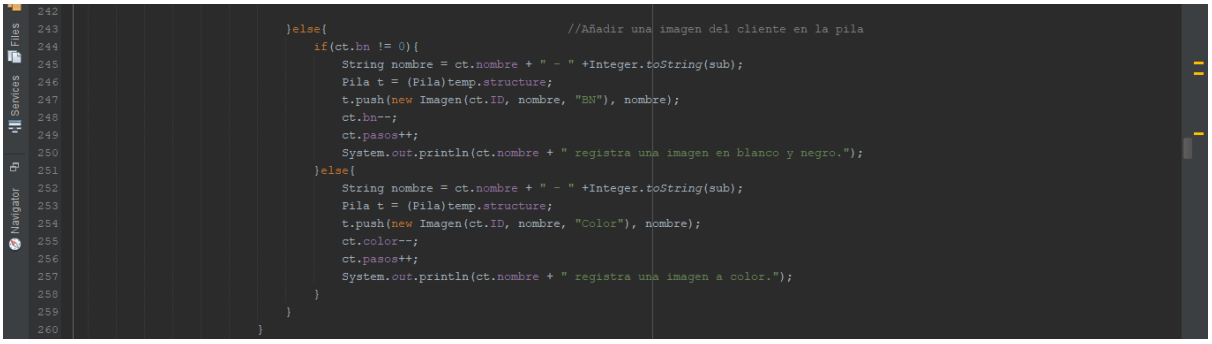
Cada ventanilla del programa maneja una pila de imágenes. También tienen a su disposición la salida de los clientes de la cola. Por motivos de optimización, la primera cosa que hacen las ventanillas es quitar a los clientes que ya finalizaron en cargar sus imágenes en el sistema. Cuando es el caso, se envían sus imágenes a las colas de impresión, este sale y entra en una sala de espera, manteniéndose ahí hasta que todas sus imágenes están impresas.



```
//Manejo de las Ventanillas-----------------------------------------------------
    //Vaciar espacios
    NodoListaPilas temp = Ventanilla.head;
    System.out.println("");
    for(int i = 0; i < Ventanilla.nc; i++){

        if(temp.content != null){   //Está llena
            Cliente ct = (Cliente)temp.content;
            int sub = ct.bn + ct.color;     //Imagenes que el cliente no ha procesado

            if(sub == 0){                       //Sacar Cliente de la ventanilla
                ct.ventana = temp.name;
                ct.pasos++;

                //Inicializa la lista para poder añadir clientes
                NodoListaDobleCircular tes = Espera.head;
                if(tes.name.equals("Vacio")){
                    Espera.deleteL();
                }

                //Añade al cliente a la lista
                ListaSimple tsi = new ListaSimple();
                tsi.add(null, "Vacio");

                Espera.add(tsi, "\"" + ct.nombre + "\"", ct);
                System.out.println(ct.nombre + " entra en la lista de espera.");
```

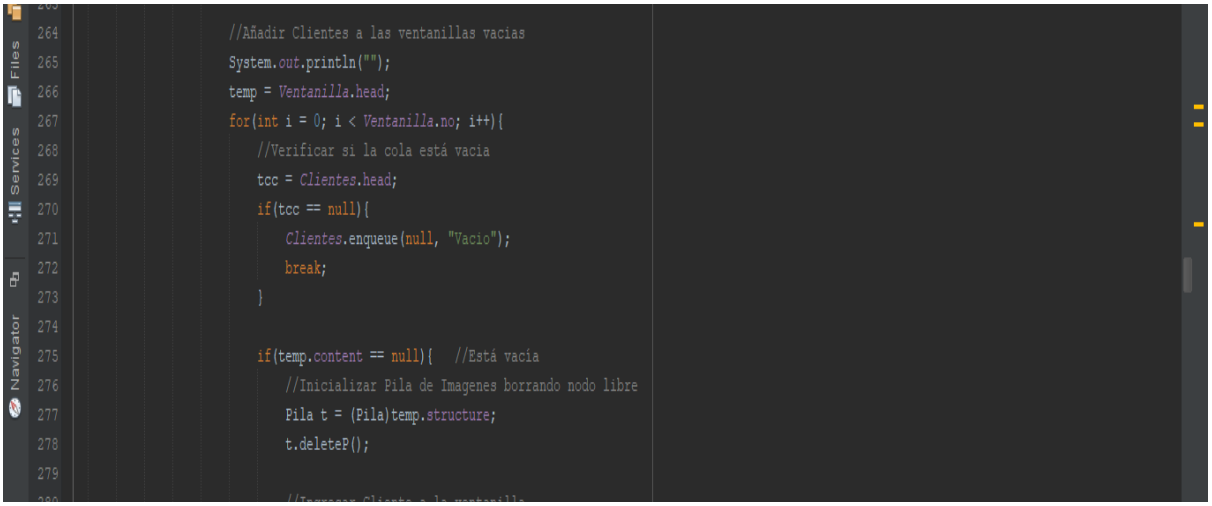
```
                    //Sacar al cliente de la ventanilla
                    temp.content = null;

                    //Vaciar Pila de imagenes
                    Pila pt = (Pila)temp.structure;
                    while(pt.head != null){
                        Imagen it = (Imagen)pt.pop();
                        if(it.Tipo.equals("Color")){
                            //Inicializar Impresoras
                            NodoCola tbn = BN.head;
                            tbn = C.head;
                            if(tbn.name.equals("Libre")){
                                C.deleteL();
                            }
                            C.enqueue(it, it.Nombre);
                        }else{
                            //Inicializar Impresoras
                            NodoCola tbn = BN.head;
                            if(tbn.name.equals("Libre")){
                                BN.deleteL();
                            }
                            BN.enqueue(it, it.Nombre);
                        }
                    }
                    System.out.println("Las " + Integer.toString(ct.total) +" imagenes de " + ct.nombre + " son enviadas a las colas de impresi
```

Si el cliente aún tiene imágenes por escanear, agrega una imagen a la pila y se va a mantener ahí hasta que termine de añadir todas sus imágenes al sistema.



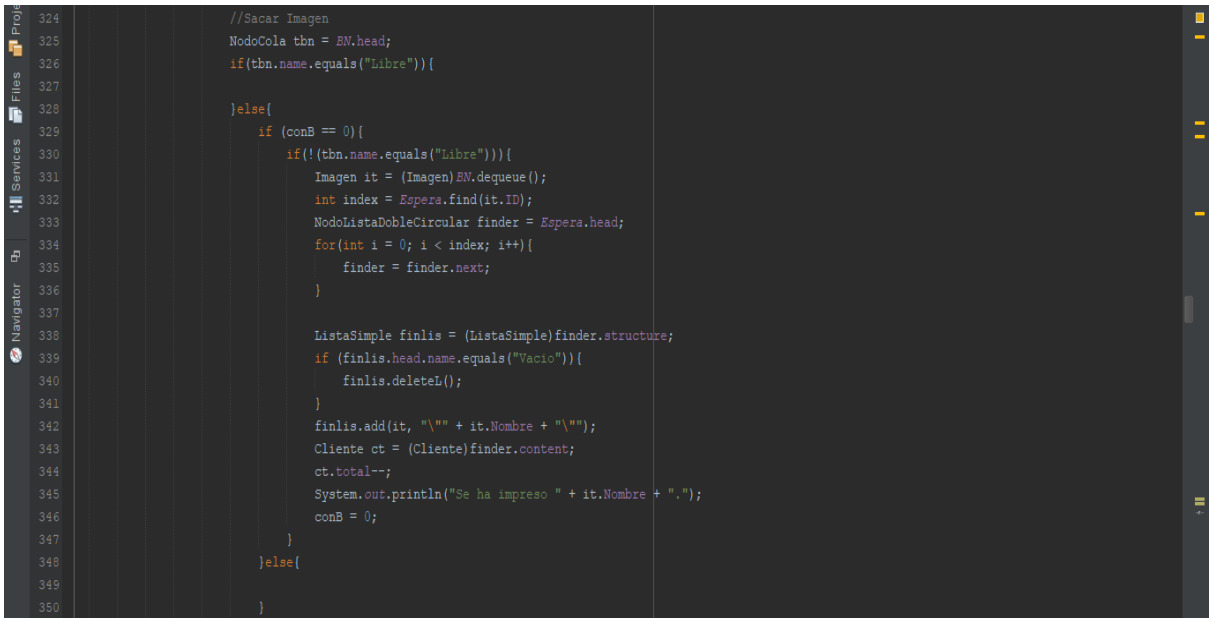
```
                }else{                              //Añadir una imagen del cliente en la pila
                    if(ct.bn != 0){
                        String nombre = ct.nombre + " - " +Integer.toString(sub);
                        Pila t = (Pila)temp.structure;
                        t.push(new Imagen(ct.ID, nombre, "BN"), nombre);
                        ct.bn--;
                        ct.pasos++;
                        System.out.println(ct.nombre + " registra una imagen en blanco y negro.");
                    }else{
                        String nombre = ct.nombre + " - " +Integer.toString(sub);
                        Pila t = (Pila)temp.structure;
                        t.push(new Imagen(ct.ID, nombre, "Color"), nombre);
                        ct.color--;
                        ct.pasos++;
                        System.out.println(ct.nombre + " registra una imagen a color.");
                    }
                }
            }
```

Con las ventanillas vacías, se pueden cargar ya los clientes. Se van recorriendo cada una para ir realizando el análisis. Al encontrar un espacio, se toma un cliente de la cola y se añade a la ventanilla vacía. Al pasar a ventanilla, el cliente añade su primera imagen en la pila de la ventanilla.



```
            //Añadir Clientes a las ventanillas vacias
            System.out.println("");
            temp = Ventanilla.head;
            for(int i = 0; i < Ventanilla.no; i++){
                //Verificar si la cola está vacia
                tcc = Clientes.head;
                if(tcc == null){
                    Clientes.enqueue(null, "Vacio");
                    break;
                }

                if(temp.content == null){   //Está vacia
                    //Inicializar Pila de Imagenes borrando nodo libre
                    Pila t = (Pila)temp.structure;
                    t.deleteP();

                    //Ingresar Cliente a la ventanilla
```

Las impresoras tiene como tarea ir procesando las imágenes. El proceso para ambas sigue el mismo concepto: Sacar imágenes de la cola de impresión. Cada imagen impresa es enviada a su cliente en la lista de espera.

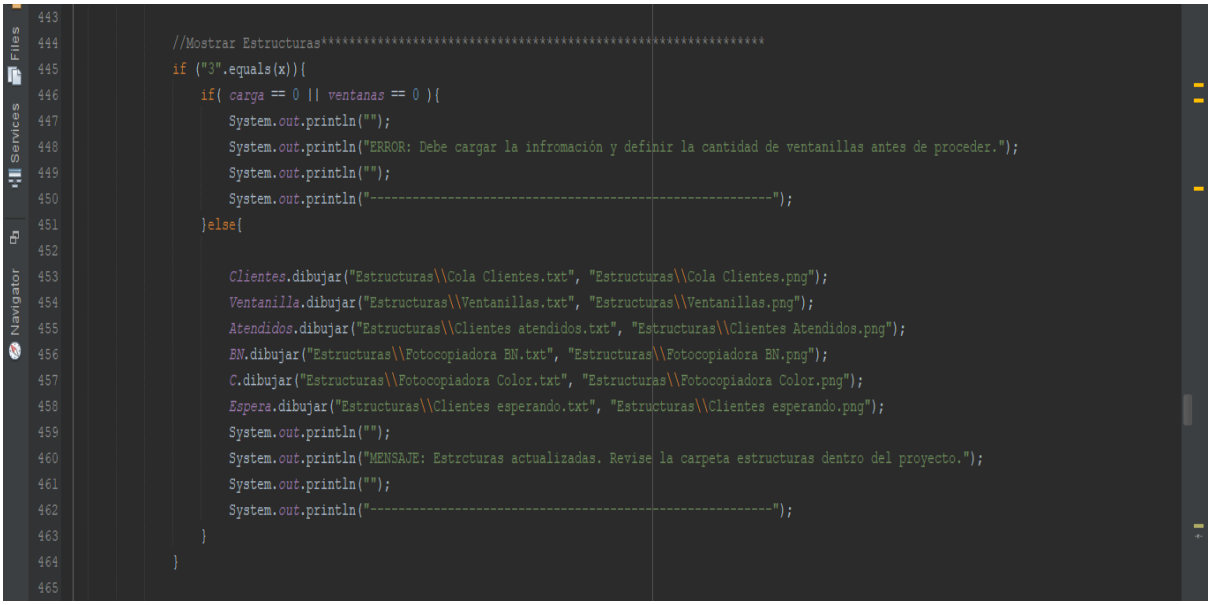


```
//Sacar Imagen
NodoCola tbn = BN.head;
if(tbn.name.equals("Libre")){

}else{
    if (conB == 0){
        if(!(tbn.name.equals("Libre"))){
            Imagen it = (Imagen)BN.dequeue();
            int index = Espera.find(it.ID);
            NodoListaDobleCircular finder = Espera.head;
            for(int i = 0; i < index; i++){
                finder = finder.next;
            }

            ListaSimple finlis = (ListaSimple)finder.structure;
            if (finlis.head.name.equals("Vacio")){
                finlis.deleteL();
            }
            finlis.add(it, "\"" + it.Nombre + "\"");
            Cliente ct = (Cliente)finder.content;
            ct.total--;
            System.out.println("Se ha impreso " + it.Nombre + ".");
            conB = 0;
        }
    }else{

    }
```

La última parte del proceso a manejar es la lista de espera. La lista de espera se encarga de revisar que los clientes en la fila sus contadores de imágenes se encuentren en cero. Cuando esto se cumple, los clientes entran en la lista de registros de la empresa donde se encuentran sus datos personales, las imágenes impresas y el número total de pasos ejecutados.

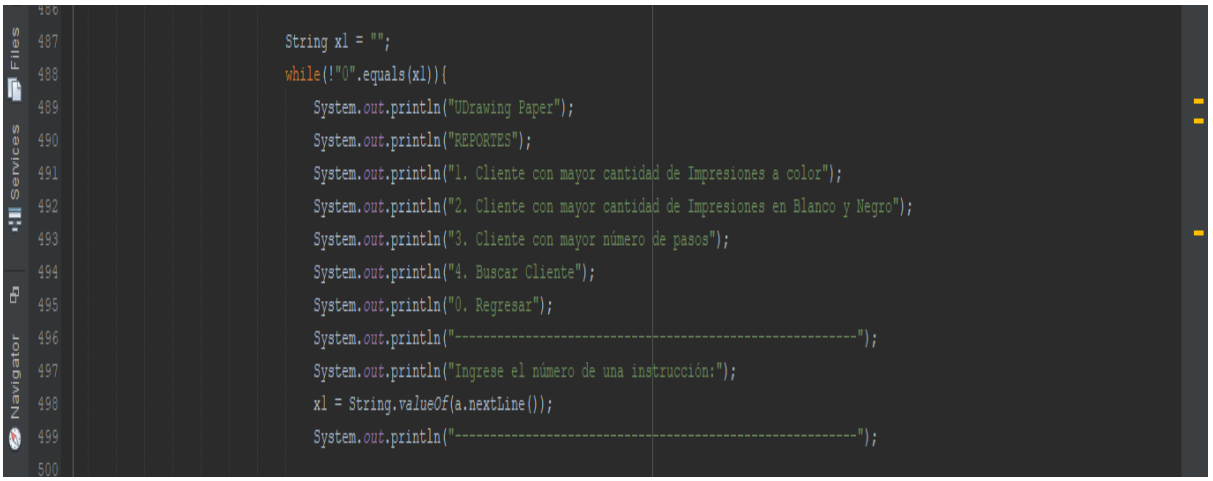
```
//Manejo de Clientes en Espera-----------------------------------------------------
    System.out.println("");
    //Verificación de Imagenes
    NodoListaDobleCircular finder = Espera.head;
    int iterador = Espera.no;
    if(!finder.name.equals("Vacio")){

        //Inicializar Lista atendidos
        NodoListaSimple lf = Atendidos.head;
        if(lf == null){

        }else{
            if(lf.name.equals("Vacio")){
            Atendidos.deleteL();
            }
        }

        for(int i = 0; i < iterador; i++){
            Cliente ct = (Cliente)finder.content;
            if (ct.total == 0){
                Atendidos.add(ct, "\"" + ct.nombre + "\"" );
                System.out.println(ct.nombre + " se ha retirado del establecimiento.");
                System.out.println("Pasos de " + ct.nombre + ": " + Integer.toString(ct.pasos));
                System.out.println("");
                Espera.remove(i);
            }else{
                ct.pasos++;
```

La tercera opción del menú principal consiste en crear los grafos de las estructuras. Cada estructura tiene implementado dentro de sus métodos el proceso de dibujo, por lo que se requiere nada más que se ejecute el comando.



```
//Mostrar Estructuras*****************************************************************
if ("3".equals(x)){
    if( carga == 0 || ventanas == 0 ){
        System.out.println("");
        System.out.println("ERROR: Debe cargar la infromación y definir la cantidad de ventanillas antes de proceder.");
        System.out.println("");
        System.out.println("-------------------------------------------------------");
    }else{

        Clientes.dibujar("Estructuras\\Cola Clientes.txt", "Estructuras\\Cola Clientes.png");
        Ventanilla.dibujar("Estructuras\\Ventanillas.txt", "Estructuras\\Ventanillas.png");
        Atendidos.dibujar("Estructuras\\Clientes atendidos.txt", "Estructuras\\Clientes Atendidos.png");
        BN.dibujar("Estructuras\\Fotocopiadora BN.txt", "Estructuras\\Fotocopiadora BN.png");
        C.dibujar("Estructuras\\Fotocopiadora Color.txt", "Estructuras\\Fotocopiadora Color.png");
        Espera.dibujar("Estructuras\\Clientes esperando.txt", "Estructuras\\Clientes esperando.png");
        System.out.println("");
        System.out.println("MENSAJE: Estrcturas actualizadas. Revise la carpeta estructuras dentro del proyecto.");
        System.out.println("");
        System.out.println("-------------------------------------------------------");
    }
}
```

La sección de los reportes posee un submenú interno. En él, es posible filtrar la información por tres atributos (pasos, imágenes a color y en B/N). También se puede realizar búsquedas de clientes atendidos usando el nombre del mismo.



```
String xl = "";
while(!"0".equals(xl)){
    System.out.println("UDrawing Paper");
    System.out.println("REPORTES");
    System.out.println("1. Cliente con mayor cantidad de Impresiones a color");
    System.out.println("2. Cliente con mayor cantidad de Impresiones en Blanco y Negro");
    System.out.println("3. Cliente con mayor número de pasos");
    System.out.println("4. Buscar Cliente");
    System.out.println("0. Regresar");
    System.out.println("-------------------------------------------------------");
    System.out.println("Ingrese el número de una instrucción:");
    xl = String.valueOf(a.nextLine());
    System.out.println("-------------------------------------------------------");
```

```
501                 //Cliente con mayor cantidad de imagenes a color
502                 if ("1".equals(xl)){
503                     System.out.println("*****************************************************");
504                     System.out.println("              TOP 5 - IMAGENES A COLOR ");
505                     System.out.println("*****************************************************");
506                     List<Cliente> orden = new ArrayList<>();
507                     NodoListaSimple to = Atendidos.head;
508                     for ( int i = 0; i < Atendidos.no; i++){
509                         orden.add((Cliente)to.content);
510                         to = to.next;
511                     }
512                     orden.sort(Comparator.comparing(Cliente::getColor).reversed());
513
514                     int iterador = orden.size();
515                     if(iterador > 5){
516                         iterador = 5;
517                     }
518                     for(int i = 0; i < iterador; i++){
519                         Cliente temp = orden.get(i);
520                         System.out.println(Integer.toString(i + 1) + ". " + temp.nombre + " - Imagenes Impresas a Color: " + temp.tc);
521                     }
522                     System.out.println("");
523                     System.out.println("-----------------------------------------------------");
524                 }
525
```

```
526                 //Cliente con mayor cantidad de imagenes en blanco y negro
527                 if ("2".equals(xl)){
528                     System.out.println("*****************************************************");
529                     System.out.println("              TOP 5 - IMAGENES EN BN ");
530                     System.out.println("*****************************************************");
531                     List<Cliente> orden = new ArrayList<>();
532                     NodoListaSimple to = Atendidos.head;
533                     for ( int i = 0; i < Atendidos.no; i++){
534                         orden.add((Cliente)to.content);
535                         to = to.next;
536                     }
537                     orden.sort(Comparator.comparing(Cliente::getNegro).reversed());
538
539                     int iterador = orden.size();
540                     if(iterador > 5){
541                         iterador = 5;
542                     }
543                     for(int i = 0; i < iterador; i++){
544                         Cliente temp = orden.get(i);
545                         System.out.println(Integer.toString(i + 1) + ". " + temp.nombre + " - Imagenes Impresas en Blanco y Negro: " + temp
546                     }
547                     System.out.println("");
548                     System.out.println("-----------------------------------------------------");
549
550                 }
551
```