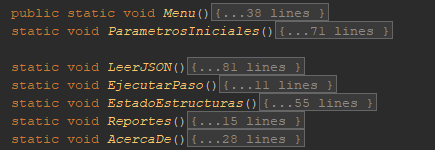
**Manual Técnico**

**Requerimientos:**

* **OS:** Windows 10 Pro 21H2
* **Procesador:** Intel(R) Core (TM) i5-4310M CPU @ 2.70GHz
* **RAM:** 8GB

**Funcionamiento del Programa:**

1. **Cosas Generales:** Se hizo uso de Programación Orientada a Objetos así como Programación Funcional mediante el uso de Métodos y Funciones para lograr la ejecución óptima del proyecto, estos son los métodos utilizados:

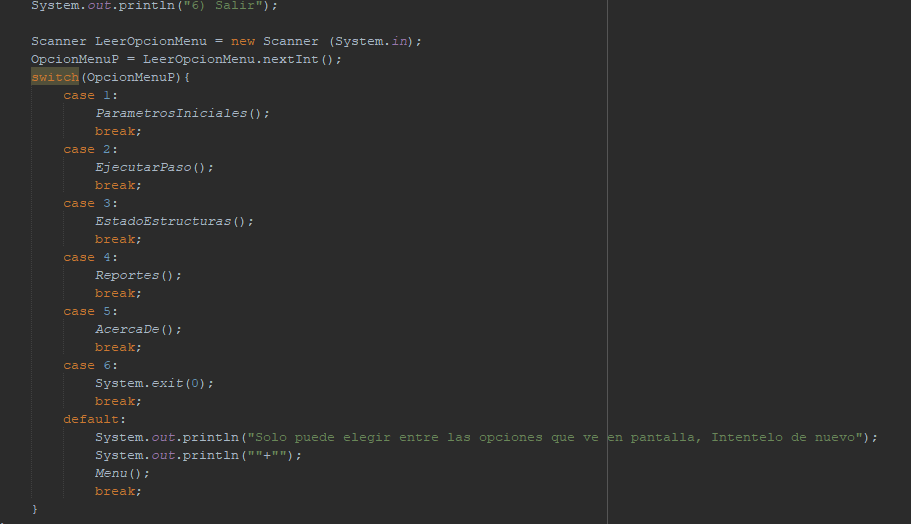
****

Y estas son las clases utilizadas:

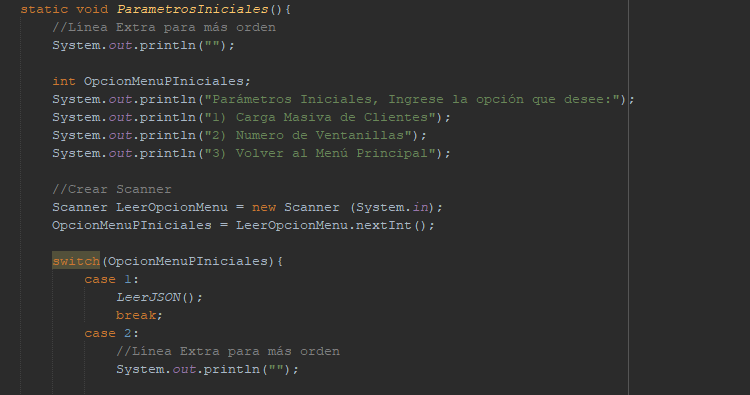


Como se puede observar es una clase por cada nodo y una por cada estructura utilizada.

1. **Menús:** Todos los menús del proyecto fueron implementados de la misma forma, desplegando las opciones en pantalla seguidas de un Scanner que lee la opción del usuario y un switch que le dice al programa qué método ejecutar:

****

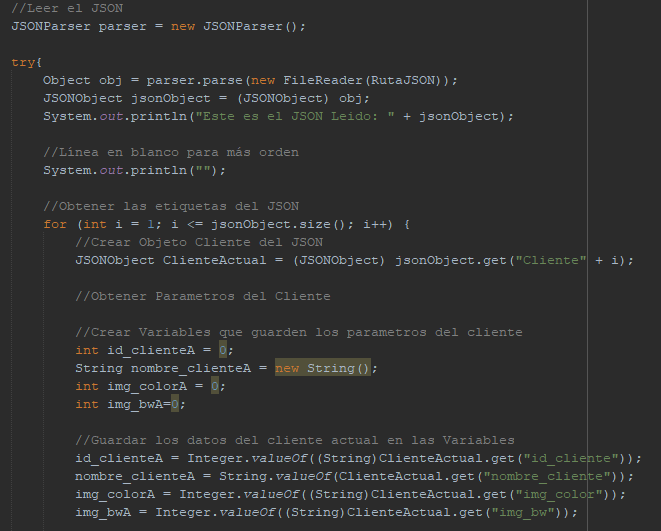
1. **Parámetros Iniciales:** Solo se muestra el menú correspondiente, en esencia, funciona igual al Menú Principal:

****

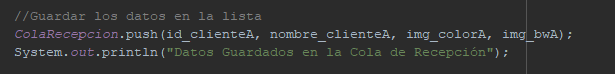
1. **Leer JSON:** Para poder leer el archivo JSON se utilizó la librería JSON Simple, la cual está instalada en las dependencias del programa:

****

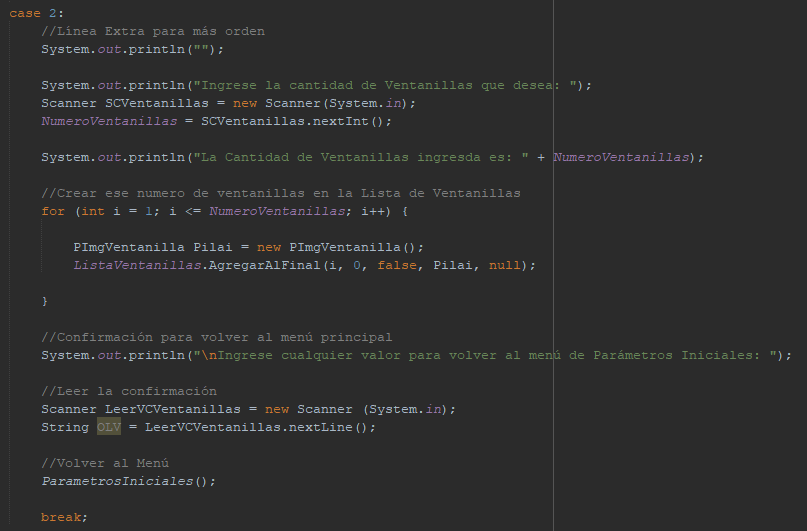
Debido a la dificultad de trabajar con Maven se creó una dependencia con un nombre arbitrario, pero el jar que contiene es la librería JSON Simple la cuál nos permite tener métodos que nos facilitan la lectura del archivo JSON, funciona de manera similar a usar ElementTree para leer archivos XML en Python:



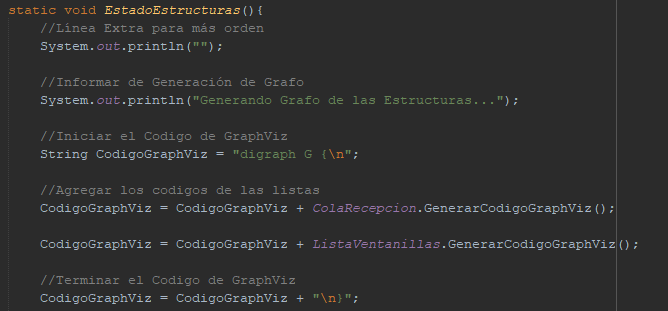
Una vez leídas las etiquetas se almacenan en la cola de Recepción, la cual veremos a fondo más adelante, así se almacenan estos datos en la lista:

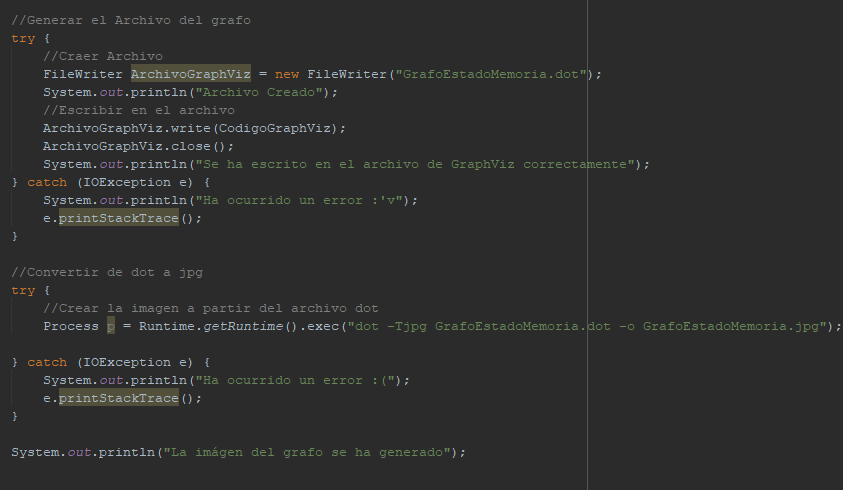


1. **Número de Ventanillas:** Para esta función se utilizó el mismo switch del menú de Parámetros Iniciales debido a que es una función corta que solo le asigna el valor que ingrese el usuario a una Variable Global llamada “NumeroVentanillas” y crea la cantidad de nodos ingresada en la lista de ventanillas. Cabe mencionar que cada Lista de Ventanillas tiene una Pila de Imágenes vacía para contener las imágenes que el cliente desee imprimir.

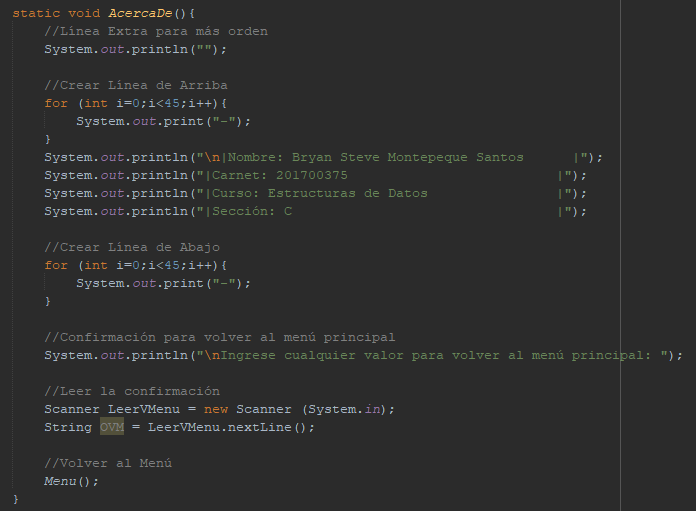
****

1. **Estado Estructuras:** Este método genera un grafo de las estructuras en tiempo real, por conveniencia el recorrer las listas es un método que se encuentra en las listas en sí, esto es debido a que por la complejidad del programa iba a ser muy laborioso y confuso recorrerlas todas en el mismo método, entonces cada lista solo genera subgrafos que contienen el cluster de sus nodos, y este método, EstadoEstructuras, solo genera las partes de código faltantes para generar el grafo completo. Así mismo genera el archivo .dot y la imagen .jpg en la dirección del proyecto:

****

Y así se generan el archivo y la imagen:

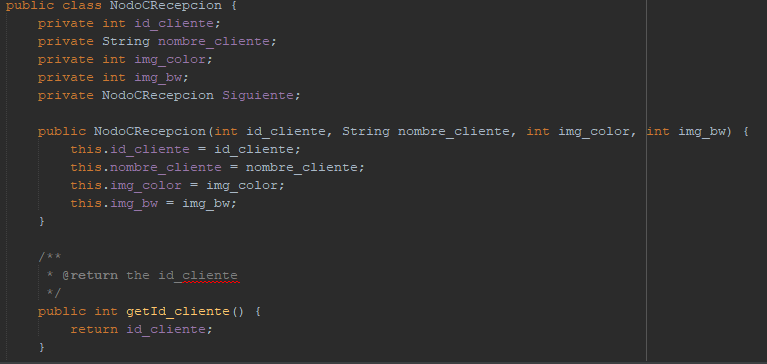
1. **Acerca De:** Solo despliega los datos del creador con sout:

****

1. **Salir:** Cierra el programa con system.exit(0);

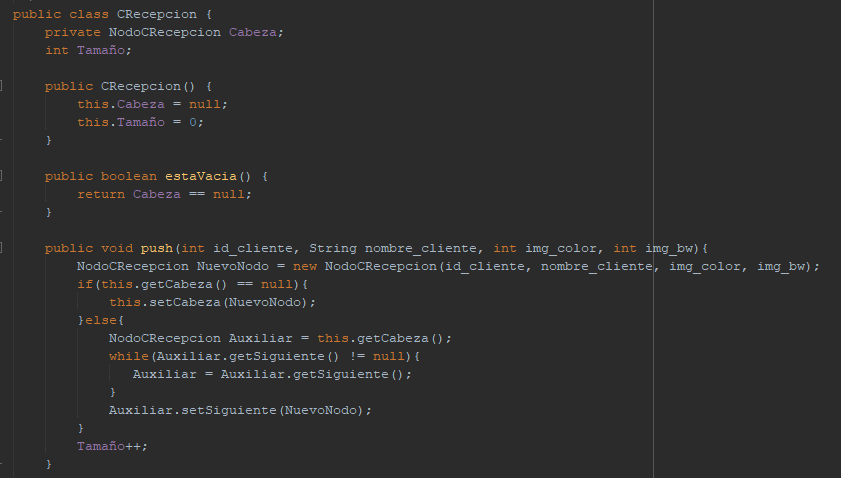
****

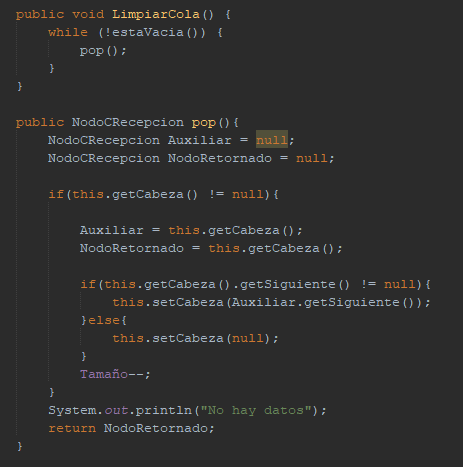
1. **Lista Simple:** Debemos empezar con los nodos, los nodos de todas las estructuras funcionan igual, son un objeto que guarda datos, debido a esto todas las clases nodo tienen la misma estructura, solo cambian sus parámetros, aquí pueden ver el nodo de la clase CRecepcion (Es decir, “ColaRecepcion”) Esta es la clase NodoCRecepcion:

****

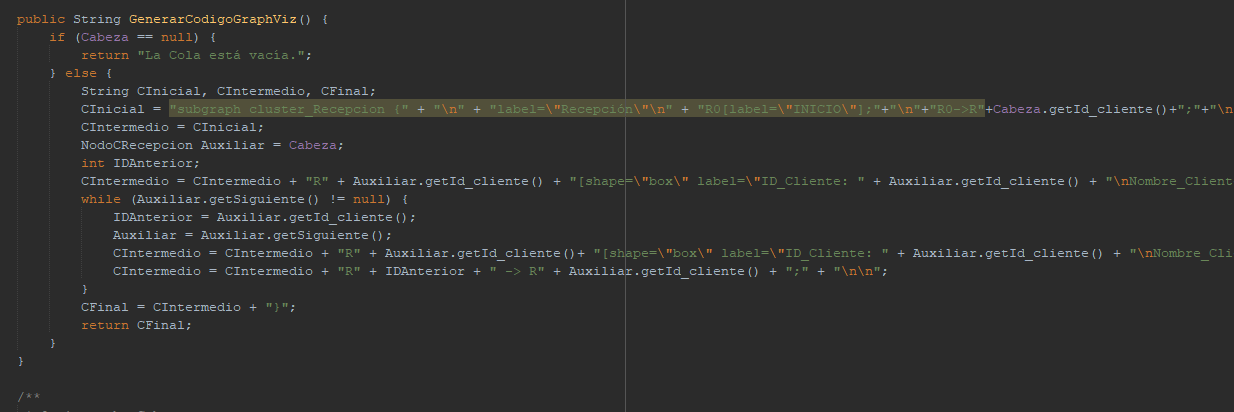
Todo lo demás de la clase de los nodos son sets y gets para poder manejarlos desde otras clases o utilizarlos con mayor facilidad.

1. **Cola Recepción:** Esta es la clase ColaRecepcion:

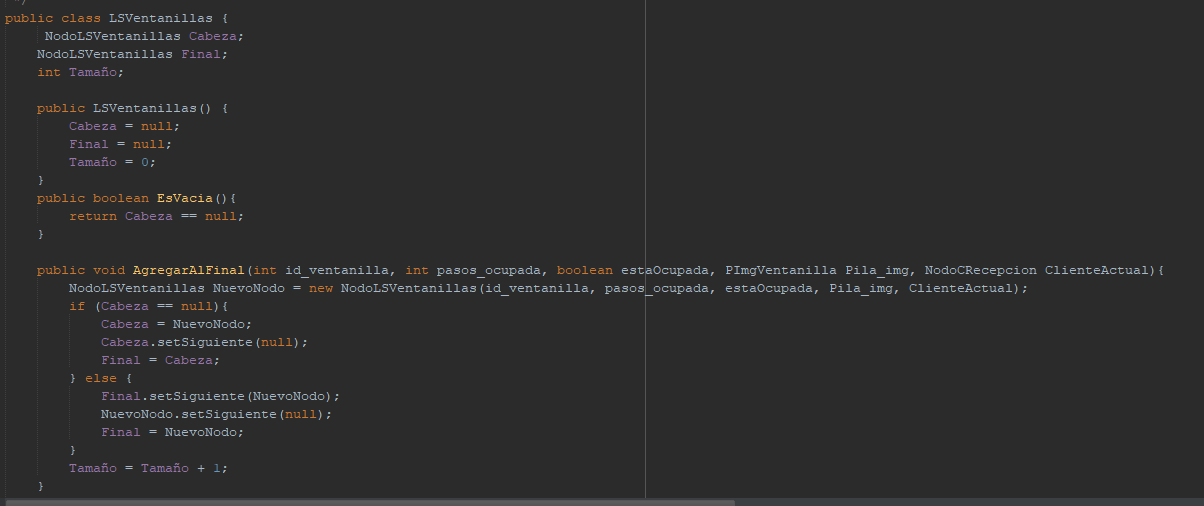
****

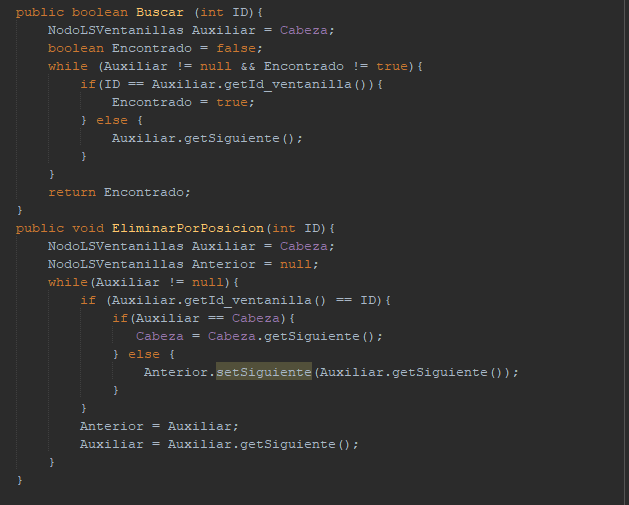
****

Cada Estructura tiene una función String como esta que recorre la lista, genera del código del Subgrafo de GraphViz y devuelve un String con ese código a donde la hayan llamdo, es gracias a estos métodos que fue más sencillo graifcar todas las estructuras en el método de EstadoEstructuras donde se grafican todas las Estructuras del programa.

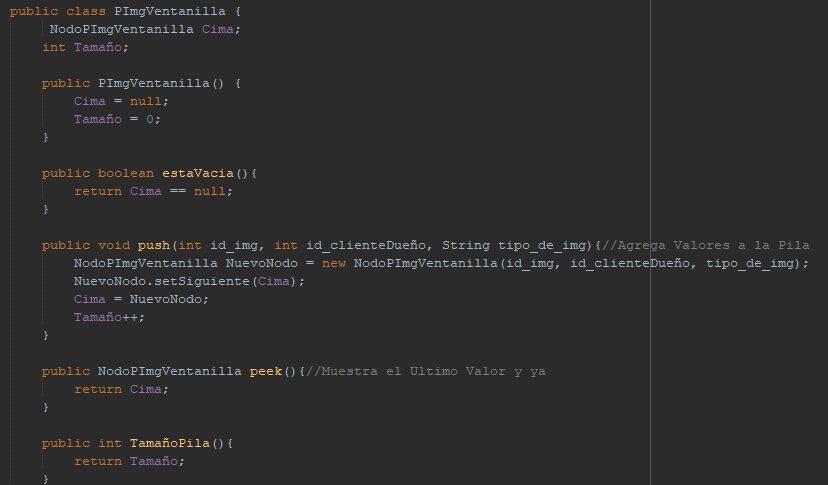
****

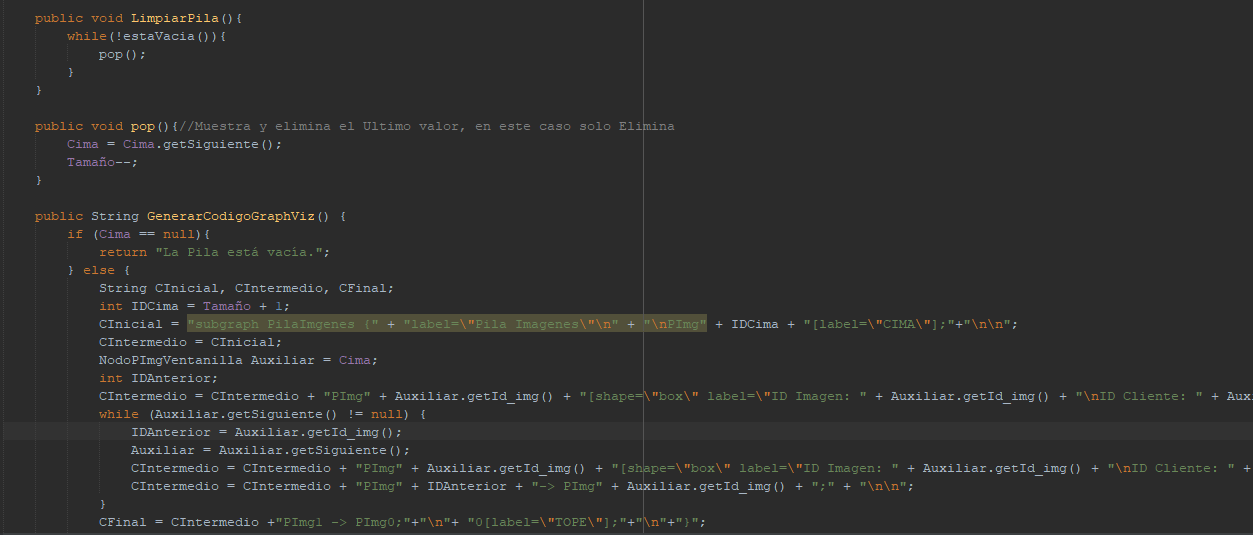
1. **Lista Simple Ventanillas:** Esta es la clase de la Lista Simple Ventanillas:

****

****

1. **Pila Imágenes:** Esta es la clase de la Pila de Imágenes que está contenida en cada Ventanilla:

****

****