Documentación del programa java

CafeteriaAppGUI Class

Campos:

cafetería: cafetería- Representa la instancia de la cafetería utilizada en la interfaz.

Campos para diferentes entradas de texto que representan varios aspectos de una cafetería como nombre del producto, porciones, precio, NIT, descripción, cantidad, etc.

Métodos:

- CafeteriaAppGUI (cafetería cafetería)- Constructor que inicializa la GUI con una referencia a la instancia de la cafetería.
- initialize (): void- Inicializa la ventana GUI y configura el diseño.
- Métodos para crear diferentes paneles:
- createAgregarProductoPanel(): JPanel- Panel para agregar un producto.
- createFacturacionPanel(): JPanel- Panel para funcionalidad de facturación.
- createInventarioPanel(): JPanel- Panel de gestión de inventario.
- createInformacionOrdenPanel(): JPanel- Panel para mostrar información del pedido.
- createEmpleadoDelMesPanel(): JPanel- Panel de gestión de datos de empleados.
- actionPerformed(ActionEvent e): void- Método ActionListener para manejar acciones de botones para registro, eliminación y actualización. Actualmente contiene lógica de marcador de posición.

Clase de CafeteriaApp

Método principal:

- main(String[] args): void- Punto de entrada de la aplicación. Crea una instancia de la clase Cafetería e inicia la GUI.

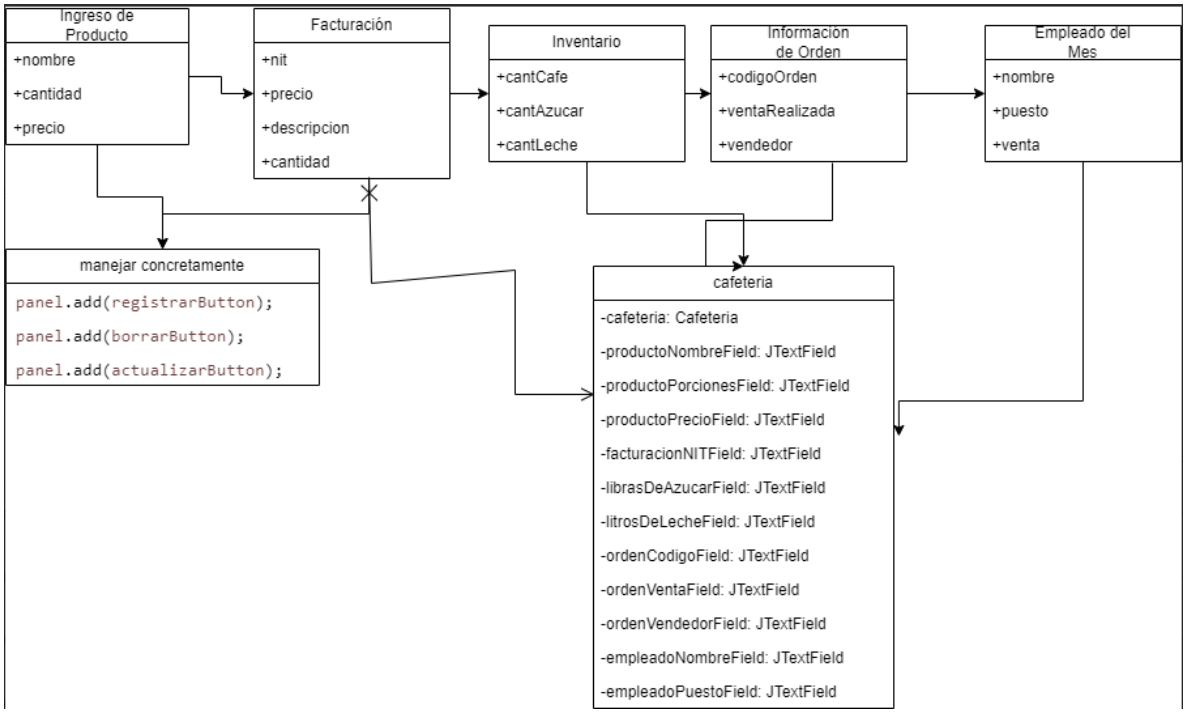
La CafeteriaAppGUIclase configura una GUI basada en Swing para una aplicación de cafetería, lo que permite a los usuarios administrar productos, facturación, inventario, información de pedidos y detalles de los empleados. La CafeteriaAppclase contiene el mainmétodo para iniciar la aplicación.

Este código crea una estructura funcional para la GUI de una aplicación de cafetería. Utiliza componentes Swing para diversas funcionalidades relacionadas con las operaciones de una cafetería. Cada panel corresponde a un aspecto específico de la gestión de la cafetería, lo que permite a los usuarios agregar productos, gestionar

la facturación, gestionar el inventario, ver los detalles de los pedidos y gestionar la información de los empleados.

Sin embargo, vale la pena señalar que el actionPerformedmétodo de la CafeteriaAppGUIclase actualmente contiene lógica de marcador de posición y requiere implementación para las funcionalidades reales como agregar, eliminar o actualizar información en la base de datos o el sistema backend.

Diagrama uml



Los patrones que use en este proyecto fue abstract factory y patrón de cadena de responsabilidad

Patrón Fábrica Abstracta:

El patrón Abstract Factory se utiliza

Implementación Sugerida:

Crear una jerarquía de fábricas para

- ProductFactory: Interfaz o clase abstracta
- EmployeeFactory: Interfaz o clase abstracta

Luego, implementa fábricas concretas:

- CoffeeFactory, SugarFactory, MilkFactory:ProductFactorypara crear productos específicos.
- BaristaFactory, ManagerFactory: Implementan EmployeeFactorypara crear diferentes tipos de empleados

Cadena de Responsabilidad de Patrón:

El patrón Chain of Responsibility se utiliza para procesar una solicitud a través

Implementación Sugerida:

Crear una cadena de responsabilidad para procesar acciones.

- Handler: Interfaz o clase abstracta que define un método para manejar la solicitud.
- RegistrarHandler, BorrarHandler, ActualizarHandler: Implementan Handlerpara manejar las acciones