María Carolina Chinchilla Rojas – 202011842

**Taller 2 – Objetos y Herencia**

El presente documento contiene una explicación de las variaciones implementadas al modelo de la lógica de la aplicación sugerido por el taller. De igual manera, resume la implementación de la consola original y la consola modificada para el tratamiento independiente de las bebidas.

PARTE 1 – LÓGICA

En la parte del módulo de la aplicación, se implementaron las clases sugeridas por el diagrama UML original junto con todos sus métodos. Adicionalmente, se realizaron cambios en la asignación de atributos de algunas de las clases. La siguiente tabla resume la implementación de las clases del módulo, así como los cambios resaltados y explicados en **negrilla**.

|  |  |
| --- | --- |
| **Clase** | **Atributos y Métodos** |
| Combo | - descuento: double  - nombreCombo: String  - **componentesCombo: List<Producto>**  **Se agregó este atributo para que todo objeto Combo contenga una lista de todos los productos que forman parte de este. De esta manera, es más sencillo calcular el precio neto y el valor del combo.** |
| + Combo(String, double)  + agregarItemACombo(Producto): void  + getPrecio(): int  + getNombre(): String  + generarTextoFactura(): String  Estas últimas tres son implementadas desde la interfaz. El precio se calcula sumando el precio individual de los componentes del combo y restando el descuento. El nombre se obtiene directamente. En la factura se escribe el texto que muestra el nombre del combo, su precio y un listado con los productos que conforman el combo. |
| Ingrediente | - nombre: String  - costoAdicional: int |
| + Ingrediente(String, int)  + getNombre(): String  + getCostoAdicional(): int |
| Pedido | - numeroPedidos: int  - idPedido: int  - nombreCliente: String  - direccionCliente: String  **- productos: List<Producto>**  **Se agregó esta clase para realizar un recorrido por los productos de un pedido de forma más sencilla.** |
| + Pedido(String, String)  + getIdPedido(): int  + agregarProducto(Producto): void  - getPrecioNetoPedido(): int  - getPrecioTotalPedido(): int  - getPrecioIVAPedido(): int  - generarTextoFactura(): String  + guardarFactura(File): void  Los métodos privados son invocados por el método guardarFactura para calcular los precios que se van a mostrar en la factura. |
| Restaurante | - pedidoEnCurso: Pedido  **- ingredientes: ArrayList<Ingrediente>**  **- menu: ArrayList<Producto>**  **- combos: ArrayList<Producto>**  **Se agregaron estos métodos para facilitar un recorrido a través de todas las opciones que ofrece el restaurante.** |
| + Restaurante()  + iniciarPedido(String, String): void  + cerrarYGuardarPedido(): void  + getPedidoEnCurso(): Pedido  + getMenuBase(): ArrayList<Producto>  + getIngredientes(): ArrayList<Ingrediente>  + cargarInformacionRestaurante(File, File, File): void  - cargarIngredientes(File): void  - cargarMenu(File): void  - cargarCombos(File): void  Los últimos tres métodos son implementados por cargarInformacionRestaurante para realizar la carga de datos. |
| <<Interface>>  Producto | No tiene atributos por ser una interfaz |
| + getPrecio(): int  + getNombre(): String  + generarTextoFactura(): String |
| ProductoAjustado | **- base: ProductoMenu**  **- adicionados: List<Ingrediente>**  **- eliminados: List<Ingrediente>**  **Se agregan los atributos porque es necesario tener una base para realizar la asignación en el constructor. Además, los adicionados y eliminados son listas de ingredientes que pueden ser recorridas.** |
| + ProductoAjustado(ProductoMenu)  + getPrecio(): int  + getNombre(): String  + generarTextoFactura(): String  Los últimos tres métodos son heredados de Producto. |
| ProductoMenu | - nombre: String  - precioBase: int |
| + ProductoMenu(String, int)  + getPrecio(): int  + getNombre(): String  + generarTextoFactura(): String  Los ultimos tres métodos son heredados de Producto. |

En todos los casos en los que se crearon atributos, fue para realizar asignaciones producto de relaciones entre clases. Por otro lado, en Restaurante y en ProductoAjustado se crearon de cero los atributos, puesto que una clase sin atributos no tiene mucho sentido; en tal situación, es más probable que no sea una clase.

PARTE 2 – CONSOLA

La consola emplea diferentes niveles de menús de interacción con el usuario.

* Menú de bienvenida
  + Consultar menú
  + Realizar pedido (Lleva al segundo menú)
  + Salir de la aplicación
* Menú de pedido
  + Añadir un producto al carrito (Lleva al tercer menú)
  + Añadir un combo al carrito (Menú estándar de combos)
  + Finalizar el pedido
  + Cancelar el pedido
* Menú de adición de producto
  + ¿Desea agregar o eliminar un ingrediente? Sí (Lleva al cuarto menú)
  + No (Menú estándar)
* Cuarto menú
  + Muestra el menú estándar para que el usuario agregue un producto al carrito
  + Luego le pide que indique el ingrediente que desea agregar o eliminar
* Menú estándar
  + Muestra los productos para que el usuario los agregue al carrito
* Menú estándar de combos
  + Muestra los combos para que el usuario los agregue al carrito

PARTE 3 – MODIFICACIONES

Se agregó un archivo de bebidas por aparte, así como una opción de cargar el archivo por aparte al del menú principal.

Se agregaron las calorías al documento para que siempre las imprima en la factura.