Taller 1 - Documentación

El objetivo del presente documento es explicar las funciones de los nuevos métodos implementados en la realización del primer taller, con respecto al esqueleto originalmente propuesto.

Parte 1 – Documentación del taller sin modificaciones

Diagrama UML:
https://app.moqups.com/xXxfQADb7Y1agg9n8YMUzzIXxb6pduh2/view/page/a1e20da75

Clase: Aplicación

En esta clase se hicieron 8 modificaciones con respecto al modelo original.

- Input(): este se encarga de recibir texto escrito por el usuario para luego ser utilizado en otras funciones que requieran el "input" del usuario.
- nuevoPedido(): recibe la información del cliente ingresada por el usuario y luego llama a la función iniciarPedido() de la clase restaurante para, justamente, inicializar un nuevo pedido.
- agregarComida(): Este método le permite al usuario agregar comida al pedido actual, dependiendo de que quiera el usuario (combo o producto del menú principal), se le mostraran el menú correspondiente y se le preguntara que seleccione alguna de las opciones disponibles y luego llama al función adiciones().
- imprimirMenu(): recibe e imprime el menú de menuProductos() de la clase Restaurante, otorgándole a cada ítem un iD para que el usuario pueda seleccionar fácilmente la opción que quiera.
- imprimirMenuCombos(): recibe e imprime el menú de Combos() de la clase Restaurante, otorgándole a cada ítem un iD para que el usuario pueda seleccionar fácilmente la opción que quiera.
- imprimirIngredientes(): (): recibe e imprime el menú de Ingredientes() de la clase Restaurante, otorgándole a cada ítem un iD para que el usuario pueda seleccionar fácilmente la opción que quiera.
- Adiciones(): Le pregunta al usuario si quiere añadir una adición o quitar un ingrediente especifico de su pedido.
- buscarPedidoPorID(): le pregunta al usuario por el iD del pedido y llama a la función encontrarPedido() de la clase Restaurante.

Clase: Restaurante

En esta clase se realizaron 15 modificaciones (7 nuevo atributos y 8 nuevos métodos) con respecto al modelo original.

- Combos: es un atributo tipo ArrayList donde se guardan todos los objetos tipo Combo para que la clase Restaurante pueda acceder a ellos fácilmente.
- Pedidos: es un atributo tipo ArrayList donde se guardan todos los objetos tipo Pedido para que la clase Restaurante pueda acceder a ellos fácilmente.
- Ingredientes: es un atributo tipo ArrayList donde se guardan todos los objetos tipo Ingrediente para que la clase Restaurante pueda acceder a ellos fácilmente.
- menuBase: es un atributo tipo ArrayList donde se guardan todos los objetos tipo productoMenu para que la clase Restaurante pueda acceder a ellos fácilmente.
- cantPedidos: es un atributo tipo Integer que guarda la información de la cantidad de pedidos que ha hecho el restaurante.
- todosPedidos: Un HashMap que guarda a los Pedidos cerrados.
- buscarProducto(String): busca un producto por su nombre en el ArrayList menuBase y lo retorna. Este se usa específicamente para la carga de datos de los combos.
- buscarItemTipoProducto(int, boolean): busca un ítem por su iD(int) depediendo su tipo (producto sencillo o combo) por medio del boolean y lo retorna. Este se utiliza específicamente para encontrar la selección del usuario.
- buscarIngredientes(int): busca un ingrediente por su iD(int) y lo retorna. Este se utiliza específicamente para encontrar la selección del usuario.
- añadirProductoAPedido(int, Pedido, boolean): a partir de la selección del usuario buscan el objeto tipo producto y lo integran a la lista del pedido actual.
- añadirQuitarIngredientePedido(Pedido, int, boolean): busca o quita un ingrediente de un producto de la elección del cliente en el pedido en curso. Se diferencia de añadirProductoAPedido() al estar editando un producto no añadiéndolo.
- encontrarPedido(int): simplemente dado un iD de un Pedido los busca en el HashMap todosPedidos y lo retorna.

Clase: Combo

Se realizo 1 modificación con respecto al modelo original.

- itemsCombo: es un ArrayList de tipo ProductoMenu donde se guardan todos los productos que hacen parte de un combo.

Clase: Pedido

Se realizaron 2 modificaciones (1 atributo y 1 método) con respecto al modelo original.

- ItemsProducto: Es un ArrayList de tipo Producto donde se guarda todos los productos (combos, productosMenu o productosAjustado) que contiene un Pedido.
- getLastProducto(): retorna el ultimo producto añadido en ItemsProducto.

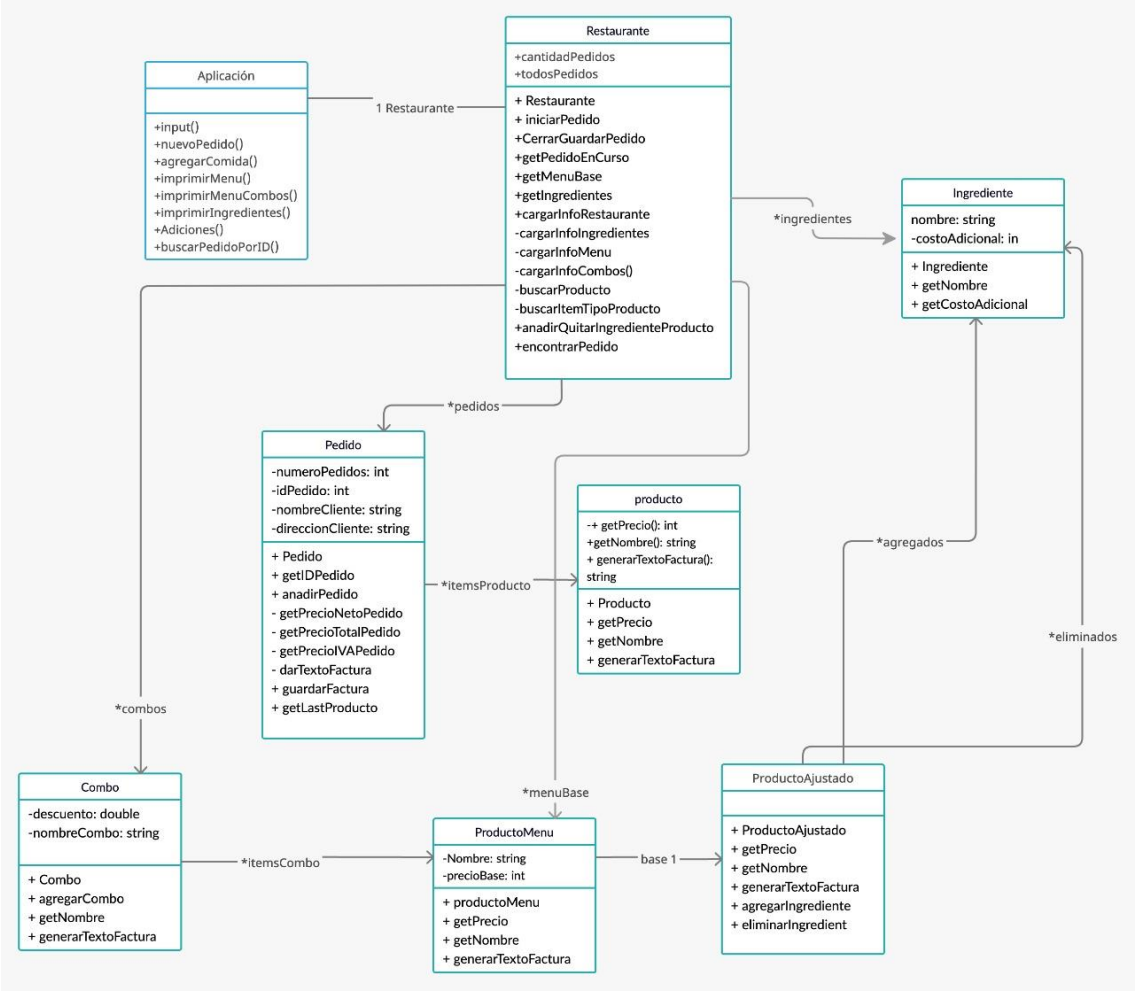
Clase: Producto Ajustado

Se realizaron 2 modificaciones con respecto al modelo original.

- agregarIngrediente(Ingrediente): Añade un Ingrediente al arreglo de agregados.
- eliminarIngrediente(Ingrediente): Añade un Ingrediente al arreglo de eliminados.

Parte 2 – Documentación del taller con modificaciones

Diagrama UML:



Clase: Restaurante

- cerrarGuardarPedido(): Revisa si hay un pedido existente idéntico al pedido en curso, en caso de ser así, imprime su ID.
- ImprimirPedidosIdenticos(pedidosIdenticos): muestra una lista de IDs con los pedidos idénticos al pedido actual.
- darPedidosIdenticos(enPedidoActual): Compara el pedido actual con lo pedidos realizados y nos indica si el actual tiene pedidos idénticos o no.
- cargarInfoBebidas(): Carga la información de las bebidas del menú.

- buscarBebida(nombreBebida): busca una bebida en la lista de Bebidas y la retorna si existe.

Clase: Bebida
- nombre: es un atributo de tipo String que en el cual guardamos el nombre de una bebida más adelante
- costoAdicional: es un atributo de tipo int que en el cual guardamos el costo de una bebida.
- calorías: es un atributo de tipo int que en el cual guardamos las calorías de una bebida.
- getNombre(): Retorna el nombre de una bebida
- getCostoAdicional(): Retorna el costo de una bebida
- getCalorias(): Retorna las calorías de una bebida

Clase: Pedido
- calorías: es un atributo de tipo ArrayList en el cual guardamos las calorías de una bebida.
- getCaloriasPedido(): calcula las calorías totales de un pedido
- comparar(otroPedido): verifica si los productos de un pedido actual son idénticos a los pedidos de uno anterior.

Interface: Producto
- calorías: es un atributo de tipo int que en el cual guardamos las calorías de una bebida.
- getCalorias(): da las calorías de un producto.

Clase: productoAjustado
- calorías: es un atributo de tipo int que en el cual guardamos las calorías de una bebida.
- getCalorias(): da las calorías de un producto ajustado.

Clase: ProductoMenu()
- calorías: es un atributo de tipo int que en el cual guardamos las calorías de una bebida.
- getCalorias(): da las calorías de un producto.

Clase: Combo()
- calorías: es un atributo de tipo int que en el cual guardamos las calorías de una bebida.
- getCalorias(): da las calorías de un combo.