**Taller 2 – Modificaciones al modelo**

**David Tobón Molina – 202123804**

**Restaurante**

Nuevos métodos:

* *getCombos()* para poder obtener información de combos cuando se imprime el menú.
* *consultarInfoPedido(String idPedido)* para poder consultar información de un pedido dado su ID. Se busca la factura (archivo .txt) que tiene como nombre el id. Así puede consultarse información de pedidos que se realizaron en otras ejecuciones del código diferentes a la actual.
* *AgregarProductobase(String nombreProductoAgregar)* para agregar la base al producto (si existe en la base de datos del restaurante) que se va a pedir e iniciar un nuevo producto ajustado con su base. Obedece al orden en que se piden los alimentos: primero la base, y después se pueden realizar las adiciones o eliminaciones de ingredientes a la base.
* *adicionarIngredienteAlProducto(String nombreIngredienteAgregar)* y *eliminarIngredienteDelProducto(String nombreIngredienteEliminar)* cuando ya se escoge una base, se le pueden añadir o quitar ingredientes.
* *confirmarProducto()* para añadir el producto ajustado al Pedido una vez ya se haya confirmado la base, y las adiciones o eliminaciones opcionales de ingredientes.
* *agregarComboAlPedido(String nombreComboAgregar)* para añadir el combo escogido al Pedido si existe en la base de datos del restaurante.
* *mostrarMenu()*: como las clase Restaurante tiene acceso a la información de los combos, productos e ingredientes, el texto del menú necesita que el restaurante pase la información de sus ítems.

Cambio de tipo de parámetros:

* Los métodos para cargar los archivos no reciben un File como parámetro, sino que reciben el String del path y nombre del archivo, y toda la lógica sobre el manejo de la clase File se hace dentro de los métodos para cargar los archivos -> la clase File no se usa por fuera de estos métodos.

**Pedido**

Nuevos atributos:

* *ArrayList<Producto> productosPedido:* como un pedido se compone de varios ítems (Productos), una ArrayList los puede guardar mientras se realiza el pedido.
* *ProductoAjustado productoActual:* como un producto ajustado consiste de una base a la cual se le agregan o quitan ingredientes, se necesita guardar la información de este tipo de producto mientras se le añaden o quitan los ingredientes a la base.

Nuevos métodos:

* *nuevoProductoAjustado(Producto nuevoItem):* se crea una instancia de un ProductoAjustado que contenga una base, este se convierte en el *productoActual* mientras se le añaden o quitan ingredientes.
* *adicionarIngredienteAlProducto(Ingrediente ingrediente)* y *eliminarIngredienteDelProducto(Ingrediente ingrediente)* para agregar o quitar Ingredientes que existen en la base de datos del restaurante al ProductoAjustado que se esté editando en el momento (guardado en el atributo de *pedidoActual)*.
* *agregarProductoAjustado():* añade al pedido el ProductoAjustado una vez se confirmen su estado final.
* *agregarCombo():* se agrega un combo de la base de datos del restaurante a la ArrayList de productos en el pedido.
* *int getIdUltimoPedido():* Como se desea guardar el número de pedidos totales, tomando en cuenta todas las ejecuciones de la aplicación, debe de haber una función que acceda a los IDs de los pedidos realizados en otras ejecuciones y obtenga la información de cuántos pedidos se han realizado con el ID mayor que encuentre en los archivos .txt.

Quitar métodos:

* Se eliminó el método agregarProducto(Producto nuevoItem)porque el ProductoAjustado se añade de manera diferente al Combo, pues uno es un atribute de esta clase, y el otro es enviado por la clase Restaurante respectivamente.

**ProductoAjustado**

Nuevos métodos:

* *adicionarIngrediente(Ingrediente ingrediente)* y *eliminarIngrediente(Ingrediente ingrediente)*: los atributos de *agregados* y *eliminados* (ArrayLists) se deben de llenar con la información necesaria, la cual es recibida por estos dos métodos en forma de un Ingrediente existente en la base de datos del restaurante.