Objetivo general del taller

El objetivo general de este taller es practicar el diseño e implementación de pruebas unitarias y de integración usando Junit.

Objetivos específicos del taller

Durante el desarrollo de este taller se buscará el desarrollo de las siguientes habilidades:

1. Diseñar un conjunto de pruebas unitarias y de integración
2. Utilizar Junit para la implementación de pruebas

Instrucciones generales

1. Descargue de Bloque Neón el archivo `Taller5-Hamburguesas_esqueleto.zip` y descomprima el archivo.
2. Importe el proyecto en Eclipse.
3. Realice las modificaciones necesarias al proyecto de acuerdo a las instrucciones que se muestran más adelante.

El taller debe desarrollarse de forma **individual**.

Descripción del caso: Restaurante de Hamburguesas

Para este taller vamos a trabajar sobre una aplicación para manejar el menú y los pedidos de un restaurante de hamburguesas. Empezaremos dándole información general sobre el caso y luego le presentaremos un modelo de clases para implementar una solución.

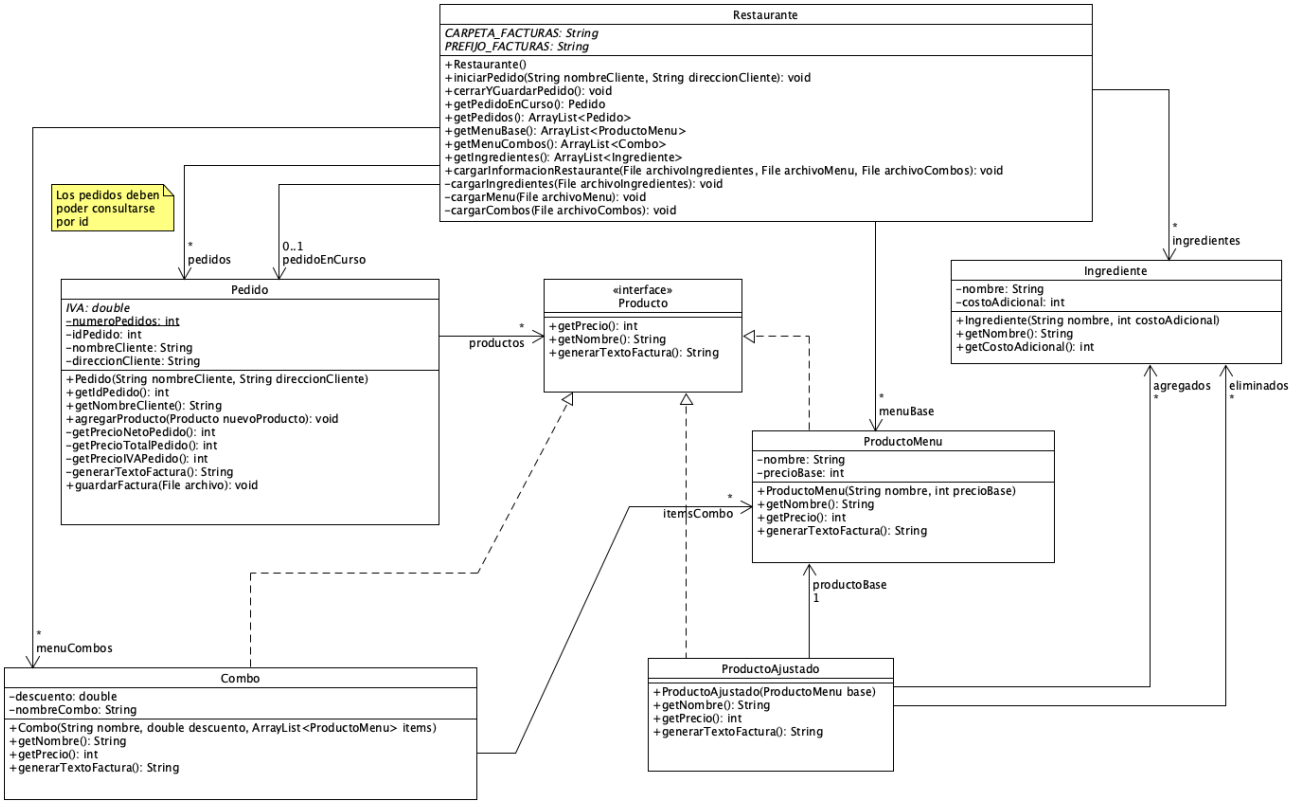
El restaurante de hamburguesas tiene un menú compuesto por unos productos básicos, combos e ingredientes adicionales. Los productos básicos son elementos que pueden hacer parte de un pedido (ej. papas fritas medianas) y cada uno tiene un precio. Los combos son agrupaciones predefinidas de productos básicos que tienen un descuento (ej. el combo especial incluye una hamburguesa sencilla y unas papas medianas y tiene un descuento del 7% sobre el precio de los productos por separado). Los ingredientes son elementos que se pueden agregar a cualquier producto o quitar de cualquier producto (ej. agregarle queso a una hamburguesa o quitarle el tomate). Los ingredientes adicionales tienen un precio adicional, pero quitar ingredientes no reduce el precio de un producto. Un combo no se puede ajustar agregándole o quitándole ingredientes.

Los clientes pueden hacer pedidos y el restaurante puede recibir un pedido a la vez (el pedido en curso). Un pedido debe tener el nombre del cliente, la dirección del cliente y los elementos que hacen parte del pedido. Esos elementos pueden ser productos básicos del menú, pueden ser combos, o pueden ser productos ajustados, es decir productos base a los cuales se les quitó o se les agregó uno o varios ingredientes.

Nota: no esperamos que la aplicación que usted construya garantice que las modificaciones a los ingredientes tengan sentido (ej. “agua sin gas con adición de suero costeño y sin cebolla” sería un elemento válido en un pedido).

Las facturas de los pedidos deben discriminar el valor de cada elemento, el valor neto total, el IVA (19%) y el valor total (neto + IVA).

El siguiente es un diagrama de clases detallado de la aplicación.



Requerimientos funcionales de la aplicación

Los siguientes son los requerimientos funcionales de la aplicación.

Funcionalidades de la aplicación

La aplicación debe ofrecer las siguientes opciones:

1. Mostrar el menú del restaurante incluyendo los productos básicos y los combos
2. Iniciar un nuevo pedido
3. Agregar un elemento a un pedido, incluyendo ajustes
4. Cerrar un pedido y guardar la factura
5. Consultar la información de un pedido dado su identificador

Cuando se abra, la aplicación debe cargar la información de los ingredientes, los productos base, y los combos de archivos de texto como los que se encuentran en la carpeta data del taller. No le debe preguntar al usuario por el nombre de estos archivos.

Cuando se cierre una factura, se debe guardar automáticamente en un archivo basado en el identificador de la factura.

Actividades: Pruebas unitarias y de integración

Con base en la documentación de la aplicación (JavaDoc), usted tendrá que diseñar e implementar pruebas unitarias y de integración utilizando el framework JUnit. Idealmente se deberían implementar primero pruebas unitarias y luego las de integración, pero el proyecto tiene comparativamente pocas clases muy acopladas entre ellas, lo cual hace difícil probarlas por separado.

Las pruebas que ustedes van a construir deben procurar que todos los métodos interesantes de las clases sean verificados: una parte importante del esfuerzo que tendrán que hacer corresponde al diseño de los escenarios de prueba y construcción de los archivos txt de prueba correspondiente.

Cuidado: el código de la aplicación tiene algunos errores. Sus pruebas deberían ser lo suficientemente buenas como para descubrirlos todos y corregirlos.

Actividades

Para cada una de las siguientes clases, diseñe los escenarios de prueba, las acciones y la forma de verificar el resultado, ANTES de empezar la implementación. Es usual que haya varios escenarios para probar una misma clase.

1. Construya una clase llamada `ProductoMenuTest` donde implemente pruebas para la clase `ProductoMenú`. Asegúrese de que sus pruebas cubran el 100% de la clase.
2. Construya una clase llamada `ProductoAjustadoTest` donde implemente pruebas para la clase `ProductoAjustado`. Asegúrese de que sus pruebas cubran el 100% de la clase.
3. Construya una clase llamada `ComboTest` donde implemente pruebas para la clase Combo. Asegúrese de que sus pruebas cubran el 100% de la clase.
4. Construya una clase llamada `PedidoTest` donde implemente pruebas para la clase `Pedido`. Asegúrese de que sus pruebas cubran el 100% de la clase.

En la clase `PedidoTest` es particularmente importante que usted verifique los métodos `generarTextoFactura` y `guardarFactura`. Tenga en cuenta esto dentro del diseño de sus escenarios de prueba.

Tenga también en cuenta que debe verificar que se lancen correctamente todas las excepciones necesarias.

Después de haber implementado las pruebas unitarias para `Producto`, `ProductoAjustado`, `Combo` y `Pedido`, construya la clase `RestauranteTest` donde implemente las pruebas para la clase `Restaurante`. Asegúrese que sus pruebas cubran la mayor cantidad posible de la clase (más del 95%).

Entrega

1. Entregue a través de Bloque Neón el URL para el repositorio de sus talleres, en la actividad designada como "**Taller 5**". El taller es individual.