**Samuel Marín- 202212565**

**EXPLICACIÓN TALLER 6**

PARTE 1

Se han construido las clases IngredienteRepetidoException y

ProductoRepetidoException que extienden la clase también construida HamburguesaException.

Las clases se han agregado al .zip del que solía ser el taller 2 bajo estos mismos nombres.

En el archivo llamado Restaurante.java se han hecho las respectivas modificaciones de los métodos cargarIngredientes

y cargarMenú para que verifiquen si hay ingredientes o productos repetidos y lancen las excepciones si es necesario.

También, se ha modificado la clase Restaurante para estar acorde a las nuevas modificaciones.

PARTE 2

En Restaurante.java se creó la clase calcularValorTotal para reportar y lanzar una excepción si el valor total

supera los 150mil pesos. Así mismo, en la clase Pedido se modificó agregarProducto para que lance una excepción

si es necesario.

Se modificó en Restaurante.java el método ConsultarPedido para que maneje la

nueva excepción agregada ValorTotalExcedidoException e imprima un mensaje

en caso de que se exceda el límite del valor total del pedido.

Además, se ha añadido "return null" en el caso de que se exceda el límite del valor total del pedido

en el método ConsultarPedido, para indicar que no se puede agregar el producto deseado al pedido.

PARTE 3

Se han creado todas las 4 clases para las pruebas unitarias, sin embargo, el código se encuentra en el

.txt bajo el nombre Pruebas Unitarias pues no me estaba dejando correr pruebas en el JUnit y no logré solucionarlo.

**Explicación Pruebas Unitarias**

**1. Construya una clase llamada ProductoMenuTest donde implemente pruebas para la clase ProductoMenú. Asegúrese de que sus pruebas cubran el 100% de la clase, sin contar los métodos getNombre y getPrecio.**

1. Se implementaron pruebas para los siguientes aspectos de la clase ProductoMenú:

El constructor: se verifica que los valores pasados al constructor sean asignados correctamente a los atributos del objeto.

El método equals(): se verifica que dos objetos ProductoMenu sean considerados iguales si tienen el mismo nombre y precio base.

El método hashCode(): se verifica que dos objetos ProductoMenu tengan el mismo código hash si tienen el mismo nombre y precio base.

El método toString(): se verifica que el resultado retornado por el método contenga el nombre y el precio base del producto.

**2. Construya una clase llamada ProductoAjustadoTest donde implemente pruebas para la clase ProductoAjustado. Asegúrese de que sus pruebas cubran el 100% de la clase, sin contar el método getNombre. Fíjese que en este caso es muy importante probar el método getPrecio.**

2. Aquí se utilizan tres métodos de prueba diferentes para cubrir distintos escenarios y verificar el método getPrecio() de la clase ProductoAjustado.

**3. Construya una clase llamada ComboTest donde implemente pruebas para la clase Combo. Asegúrese de que sus pruebas cubran el 100% de la clase, sin contar los métodos getNombre y getPrecio.**

3. Aquí se utilizan tres métodos de prueba diferentes para cubrir distintos escenarios y verificar el método getPrecio() de la clase Combo.

**4. 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 lance correctamente la excepción cuando el valor total de un pedido supere los 150.000 pesos.**

4. En el método testGenerarTextoFactura(), se crea un objeto Pedido y se agregan productos a él. Luego, se verifica si el texto de la factura generado por el método generarTextoFactura() coincide con el texto de factura esperado.

En el método testGuardarFactura(), se crea un objeto Pedido, se agregan productos a él y se guarda la factura en un archivo. Luego, se verifica si el archivo de factura fue creado correctamente.

En el método testCalcularValorTotalExcedidoException(), se crea un objeto Pedido y se agregan productos que exceden el valor total permitido. Se verifica que se lance la excepción ValorTotalExcedidoException cuando se llama al método calcularValorTotal().