Gabriel Padilla 202224552

Documento Taller 2 D.P.O.O

Parte 1: Justificación de desviaciones

Vamos a ir clase por clase, justificando y explicando las discrepancias entre los métodos propuestos en el diagrama UML y los que fueron implementados.

- Pedido:
  - Diferencias:
    - estaCerrado(): No está en el UML pero está en el código.
    - agregarProductoAjustado(ProductoAjustado productoAjustado): No está en el UML pero está en el código.
    - agregarCombo(Combo combo): No está en el UML pero está en el código.
    - getProductos(): No está en el UML pero está en el código.
    - getProductosAjustados(): No está en el UML pero está en el código.
    - cerrar(): No está en el UML pero está en el código.
    - getNombreCliente(): No está en el UML pero está en el código.
    - getDireccionCliente(): No está en el UML pero está en el código.

  - Justificación:
    - Estos métodos son útiles para operaciones más específicas y detalladas con el Pedido, lo que justifica su inclusión en la implementación aunque no estén en el diseño inicial del UML. También cabe resaltar que los métodos y atributos incluidos en el diagrama también están en el código, el único cambio acá fue que se agregaron métodos pero no se le quitaron por lo que no debería haber ningún problema.

- Combo:
  - Diferencias:
    - agregarItemACombo(ItemCombo: Producto): Está en el UML pero no está en el código.
    - toString(): No está en el UML pero está en el código.

  - Justificación:
    - La función toString() es sobrescrita para dar una representación string de un objeto, lo que justifica su inclusión.
    - agregarItemACombo(ItemCombo: Producto): Este método no fue incluido ya que en las instrucciones dice explícitamente que a un combo no se le pueden agregar Ítems:

      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.

Gabriel Padilla 202224552

-Ingrediente:

Diferencias:

getCostoAdicional(): Está en el UML como getCostoAdicional(), pero en el código está como getPrecio().

toString(): No está en el UML pero está en el código.

Justificación:

La función toString() es sobrescrita para dar una representación string de un objeto.

El cambio de nombre de getCostoAdicional() a getPrecio() mantiene la nomenclatura coherente con otras clases que tienen precios.

-ProductoAjustado:

Diferencias:

agregarIngrediente(Ingrediente ingrediente): No está en el UML pero está en el código.

eliminarIngrediente(Ingrediente ingrediente): No está en el UML pero está en el código.

getProductoBase(): No está en el UML pero está en el código.

getIngredientesAdicionales(): No está en el UML pero está en el código.

getPrecioTotalConIngredientes(): No está en el UML pero está en el código.

toString(): No está en el UML pero está en el código.

Justificación:

Estos métodos adicionales facilitan la manipulación y visualización de un ProductoAjustado, lo que justifica su inclusión. También cabe resaltar que los métodos y atributos incluidos en el diagrama también están en el código, el único cambio acá fue que se agregaron métodos pero no se le quitaron por lo que no debería haber ningún problema.

-ProductoMenu:

Diferencias:

toString(): No está en el UML pero está en el código.

Justificación:

La función toString() es sobrescrita para dar una representación string de un objeto. También cabe resaltar que los métodos y atributos incluidos en el diagrama también están en el código, el único cambio acá fue que se agregaron métodos pero no se le quitaron por lo que no debería haber ningún problema.

-Restaurante:

Diferencias:

En el código existen los siguientes métodos que no están en el UML:

agregarProductoAlPedido(Producto producto)

agregarProductoAjustadoAlPedido(ProductoAjustado productoAjustado)

agregarComboAlPedido(Combo combo)
En el UML, los métodos cargarInformacionRestaurante(), cargarIngredientes(), cargarMenu(), y cargarCombos() tienen parámetros, pero en el código no los tienen.

Justificación:

Los métodos adicionales en el código (agregarProductoAlPedido, agregarProductoAjustadoAlPedido, agregarComboAlPedido) son útiles para operaciones más específicas y detalladas con el Pedido, lo que justifica su inclusión en la implementación aunque no estén en el diseño inicial del UML.
La diferencia en los parámetros de los métodos de carga ocurre ya que en la carga individual de cada elemento (combos, ingredients y menu) ya se tienen en cuenta el nombre del archivo por lo que no es necesario tener el metodo cargarInformacionRestaurante()

-Aplicación:

Diferencias:

En el código, hay un método iniciar() que no está presente en el UML. Este es una función principal para arrancar la aplicación en lugar del método main.
En el código hay los siguientes métodos que no están en el UML:

- agregarProductoAlPedido()
- agregarProductoAjustadoAlPedido()
- agregarComboAlPedido()
- finalizarPedido()

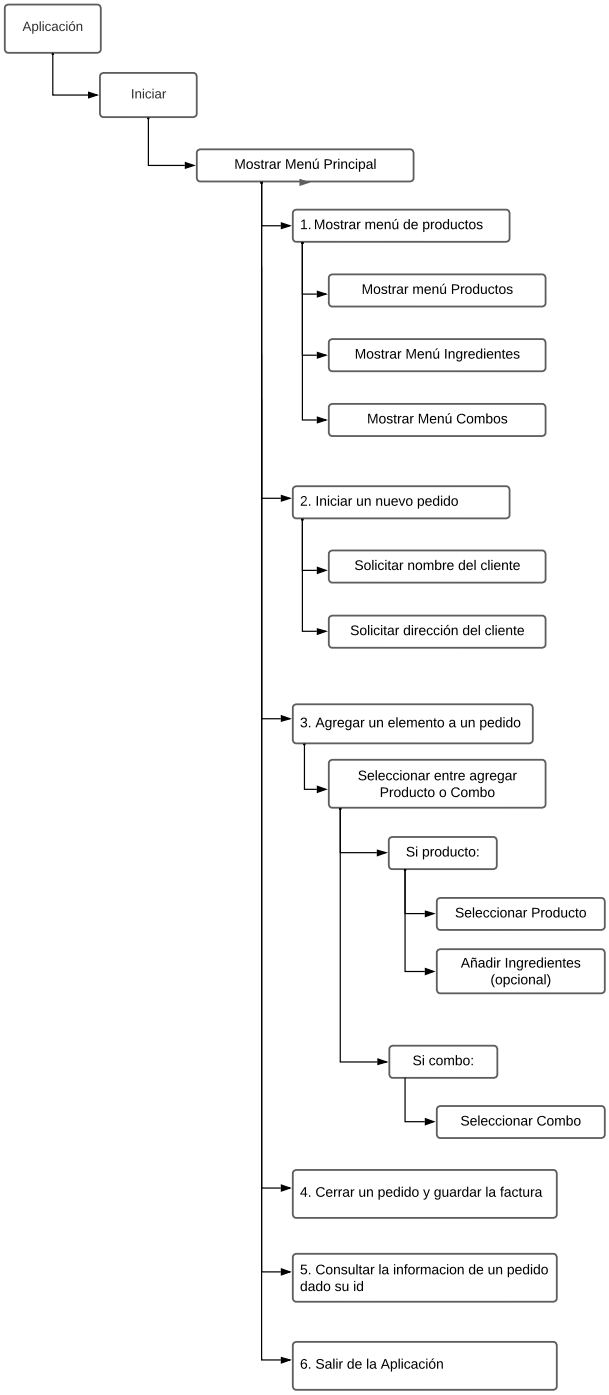
Justificación:

El método iniciar() en el códigoa inicia la aplicación, lo que puede ser más intuitivo que llamar directamente a main.
Los métodos agregarProductoAlPedido, agregarProductoAjustadoAlPedido, agregarComboAlPedido y finalizarPedido en el código proporcionan una forma más detallada de interactuar con el pedido desde la aplicación. Esto indica que, en la implementación, se dan más funcionalidades directamente en la interfaz de usuario (la clase Aplicacion).

Gabriel Padilla 202224552

Parte 2: Justificación de desviaciones

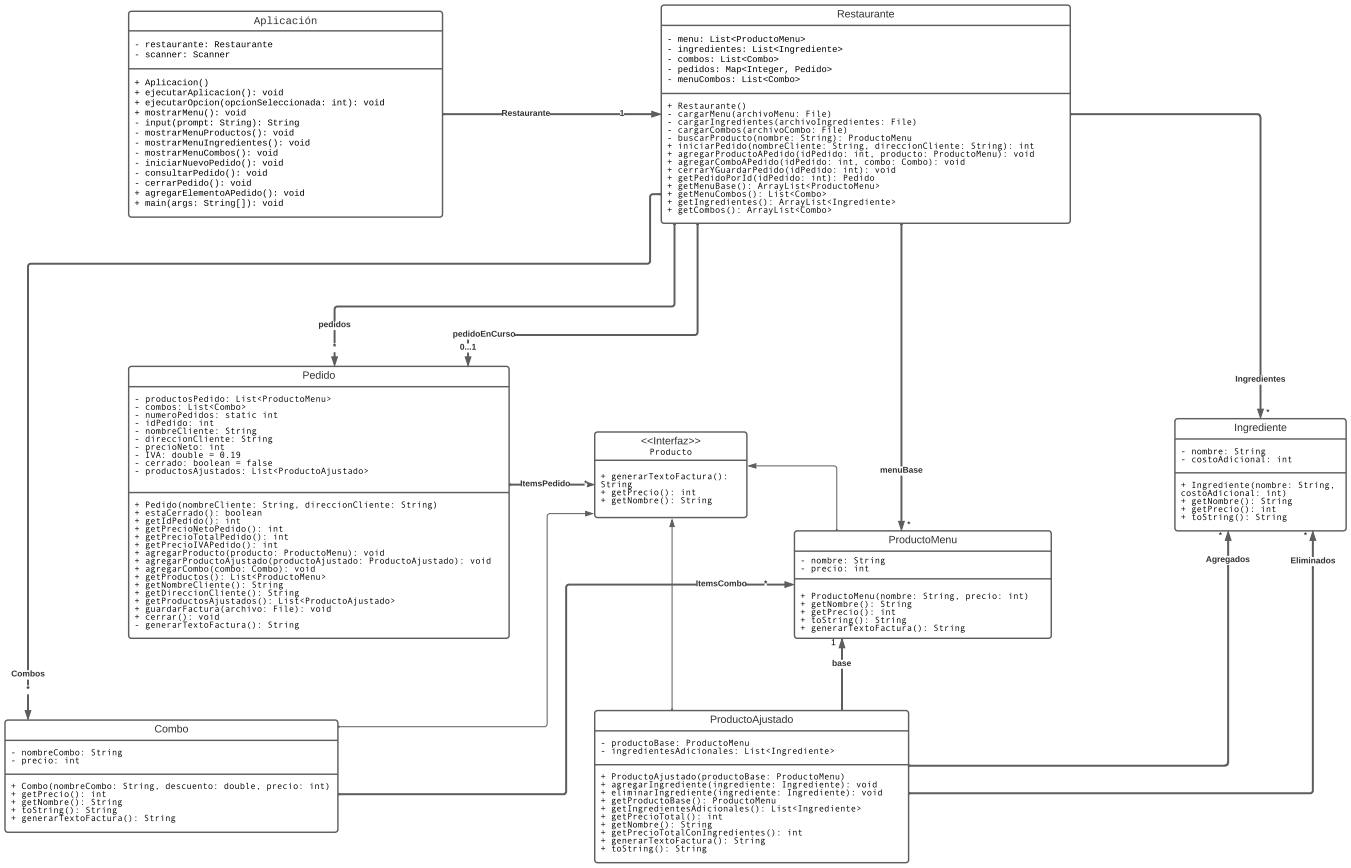
1.



Gabriel Padilla 202224552

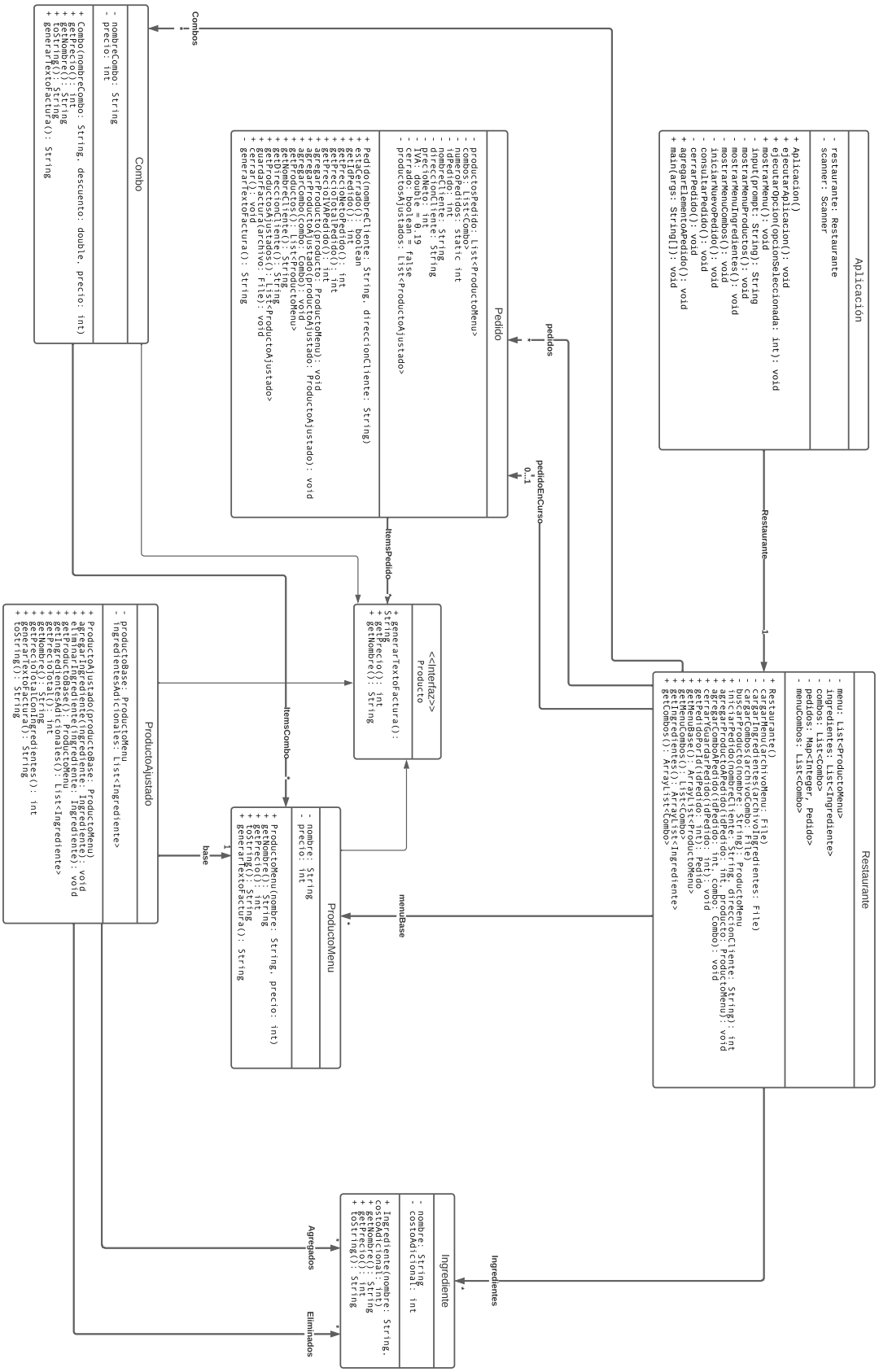
Gabriel Padilla 202224552

2.



En caso de que no se alcance a leer el UML, lo insertaré de manera horizontal en la siguiente página

Gabriel Padilla 202224552



Gabriel Padilla 202224552