**APP**

-CrearPedido():

Cuando se crea un pedido, se envían los siguientes parámetros:

Cliente, Negocio, Arreglo de Productos, Arreglo de cantidades y comentario

De esta manera, se accede al cliente y se invoca el método CrearPedido de Cliente de la siguiente manera: cliente.CrearPedido(negocio,comentario)

Una vez creado el pedido se debe agregar los productos al pedido utilizando los vectores obtenidos desde la app. Para eso hacemos uso del método AgregarProductos de la clase Cliente enviándole los siguientes parámetros:

Pedido, Arreglo de Productos, Arreglo de Cantidades.

cliente.AgregarProductos(autonumérico,Arreglo de productos, Arreglo de Cantidades)

**CLIENTE**

-AgregarProductos(): En este método se accede al pedido recibido por parámetro para poder invocar el método AgregarProducto de Pedidos de manera iterativa hasta agregar todos los productos. Es decir, el método AgregarProductos de Cliente debe acceder al método AgregarProducto de Pedido de manera iterativa.

**PEDIDO**

-AgregarProducto(): Este método se encarga de crear un DetalleProducto e insertarlo a la lista de detallesProductos en la clase Pedido.

-CalcularMontoTotal():

**ID de pedido y Id de Cliente**

El idPedido debe ser único en todo el sistema,

**RELACION de NEGOCIO y PEDIDO**  
La técnica de usar una declaración anticipada (**forward declaration**) en un archivo de encabezado se utiliza para declarar la existencia de una clase o tipo antes de proporcionar su definición completa. Esto se hace utilizando la palabra clave **class** seguida del nombre de la clase, sin proporcionar su implementación completa.

Esta técnica ayuda a evitar ciclos de dependencia entre archivos de encabezado, permitiendo que el compilador reconozca el nombre de la clase para su uso en ese contexto, sin la necesidad de incluir la definición completa de la clase.

En resumen, el uso de la declaración anticipada es una forma de proporcionar una referencia hacia adelante de un tipo o clase sin necesidad de detallar toda su implementación, lo que ayuda a evitar problemas de dependencia circular entre archivos de encabezado.

En el caso específico que mencionaste, al utilizar **class Negocio;** en el archivo de encabezado **Pedido.h**, estás empleando esta técnica para informar al compilador sobre la existencia de la clase **Negocio** antes de proporcionar su implementación completa, lo que evita el ciclo de dependencia entre los archivos de encabezado.