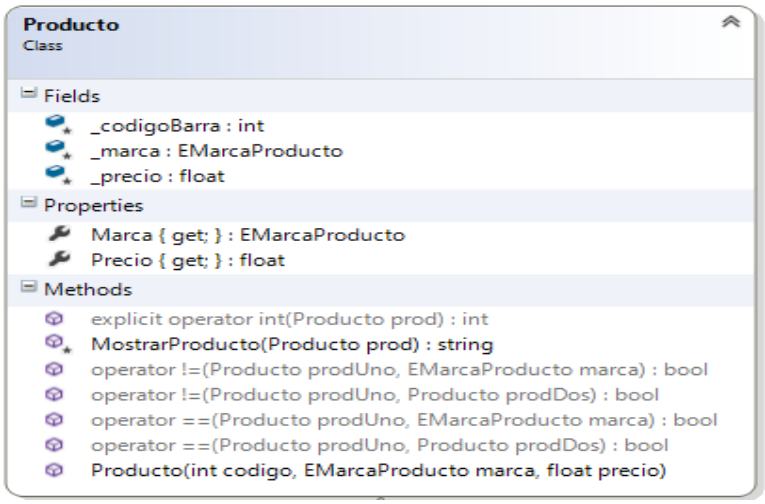
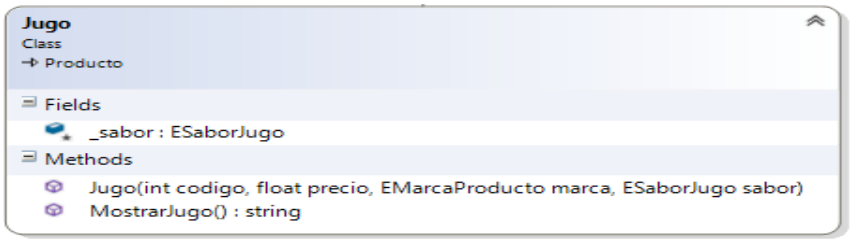
Generar una Solución nombrada como: *Apellido.Nombre.Division*, que contenga un proyecto de tipo *Biblioteca de Clases* (**Entidades**) el cual tendrá la clase base **Producto**.

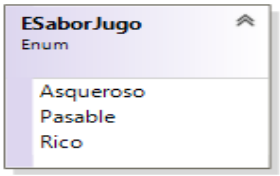
Universidad Tecnológica Nacional **Facultad Regional Avellaneda**										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: LABORATORIO II										
Apellido:				**Fecha:**						
Nombre:				**Docente(2):**						
División:				**Nota(2):**						
Legajo:				**Firma(2):**						
Instancia(1):	**PP**	**X**	**RPP**		**SP**		**RSP**		**FIN**	

1) Todos sus atributos son protegidos. Posee **sólo** un constructor de instancia. La propiedad **Marca**, retornará el valor correspondiente del atributo *_marca*. La propiedad **Precio**, retornará el valor asociado al atributo *_precio*. El método de **clase MostrarProducto**, que es protegido, retornará una cadena detallando los atributos de la clase.
2) La clase Producto posee sobrecarga de operadores:
   Igualdad (Producto, Producto). Retornará *true*, si las marcas y códigos de barra son iguales, *false*, caso contrario.
   Igualdad (Producto, EMarcaProducto). Retornará *true*, si la marca del producto coincide con el enumerado pasado por parámetro, *false*, caso contrario.
   Explícito. Retornará el código de barra del producto que recibe como parámetro.



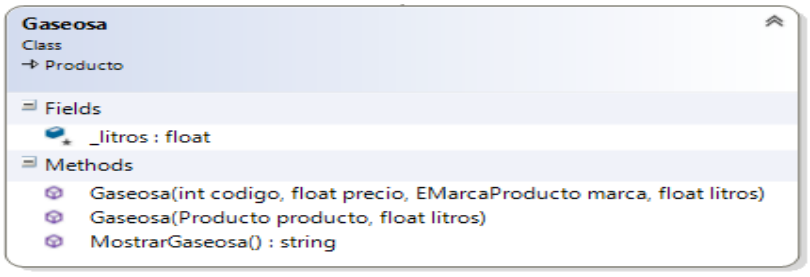
También tendra las siguiente clases derivadas de producto:





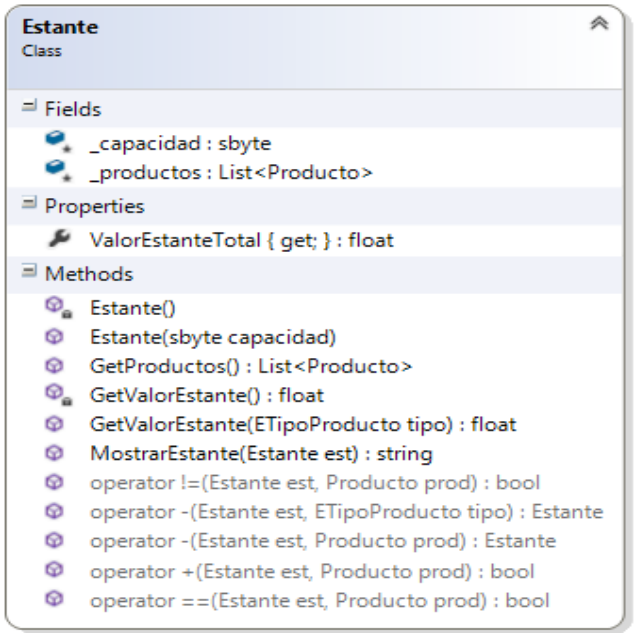
3) Jugo posee un único atributo propio, que será inicializado por su **único** constructor.
   El método público de instancia **MostrarJugo**, retornará una cadena conteniendo la información completa del objeto. Reutilizar código.

4) Galletita posee un único atributo propio, que será inicializado por su **único** constructor.
   El método público de clase **MostrarGalletita**, retornará una cadena conteniendo la información completa del objeto recibido por parámetro. Reutilizar código.



5) Gaseosa posee un único atributo propio, que será inicializado **sólo** por una de las sobrecargas del constructor (reutilizar código). El método público de instancia **MostrarGaseosa**, retornará una cadena conteniendo la información completa del objeto. Reutilizar código.

La última clase que tendrá el proyecto será **Estante**. Dicha clase posee dos atributos, ambos protegidos. Uno indicará la capacidad máxima que tendrá el estante para almacenar productos. El otro es una colección genérica de tipo Producto.



6) El constructor de instancia **privado** será el **único** que inicializará la lista genérica. La sobrecarga pública del constructor inicializará la capacidad del estante. Reutilizar código.

7) El método público GetProductos, retornará el valor asociado del atributo *_productos*.

El método público de **clase MostrarEstante**, retornará una cadena con toda la información del estante, incluyendo el detalle de cada uno de sus productos. Reutilizar código.

8) Sobrecarga de operadores:

Igualdad, retornará *true*, si es que el producto ya se encuentra en el estante, *false*, caso contrario.

Adición, retornará *true*, si el estante posee capacidad de almacenar al menos un producto más y dicho producto no se encuentra en el estante, *false*, caso contrario. Reutilizar código.

Sustracción (Estante, Producto), retornará un estante sin el producto, siempre y cuando el producto se encuentre en el listado. Reutilizar código.

Sustracción (Estante, ETipoProducto), retornará un estante con todos los productos menos el que coincida con el enumerado que recibe como parámetro. Reutilizar código.

Ejemplo: estanteSinJugo = estante – EtipoProducto.Jugo;

9) Método público y de instancia **GetValorEstante**, retornará el valor del estante de acuerdo con el enumerado que recibe como parámetro.

Ejemplo: precioGalletitas = estante.GetValorEstante(ETipoProducto.Galletita);
//Retorna sólo el valor de la suma de las galletitas

La propiedad pública **ValorEstanteTotal** está asociada a la sobrecarga privada y de instancia del método GetValorEstante. Reutilizar código.

10) Agregar a la solución un proyecto de tipo Aplicación de Consola (**TestEstante**) y agregar un método de clase que permita ordenar la lista de productos del estante a través del método Sort de dicha lista genérica. Agregar el Main (que le entregará el docente) sin modificar línea alguna.