Universidad Tecnológica Nacional

Facultad Regional Avellaneda

Modelo Primer Parcial Lab II – A321 – 2024

Criterios de evaluación

- Se anulará el puntaje si hay 2 errores en el mismo tema.
- Sus datos personales deben estar en el nombre de la carpeta principal y la solución: *Apellido.Nombre.PP. Ejemplo: Pérez.Juan.PP*. No se corregirán proyectos sin autor identificable.
- No se corregirán exámenes que no compilen.
- No se corregirán exámenes que posean commits luego de la finalización del parcial.

Condiciones de aprobación

- Respetar todas las consignas dadas.
- Todas las clases, métodos, atributos, propiedades, etc. Sean nombradas exactamente como fue pedido en el enunciado.
- Se introducta el código de la función Main sin modificaciones.
- El proyecto no contenga errores de ningún tipo.
- El código compile y se ejecute de manera correcta.
- El código no se encuentre repetido con otro compañero (quedan anulado ambos parciales).
- La salida por pantalla sea copia fiel de la entregada en este mismo documento.
- Se deberá reutilizar código cada vez que se pueda, aunque no este explícitamente en el contenido del texto.
- Se deberá documentar el código según las reglas se estilo de la catedra.
- Colocar nombre de la clase (en estáticos), this o base en todos los casos que corresponda.
- Se debe aplicar los principios de los 4 pilares de la POO.

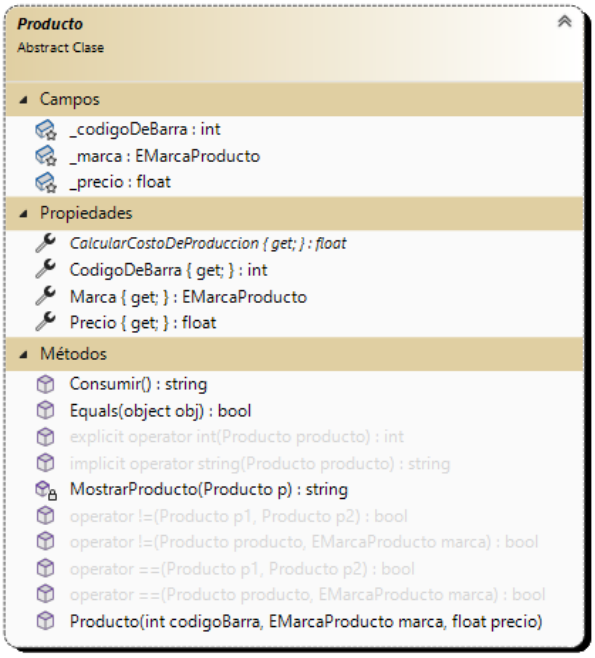
Forma de entrega

- Se deberá subir al repositorio de GitHub declarado.
- Dentro de una carpeta llamada PP.
- Deberá estar correctamente publicado y accesible para su descarga por cualquier método.

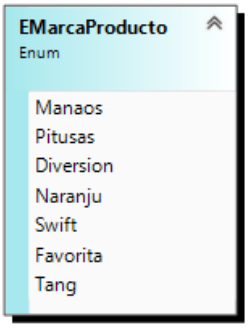
Consigna

Generar un proyecto de **tipo Biblioteca de Clases** (Entidades) el cual tendrá las siguientes clases:

Producto:



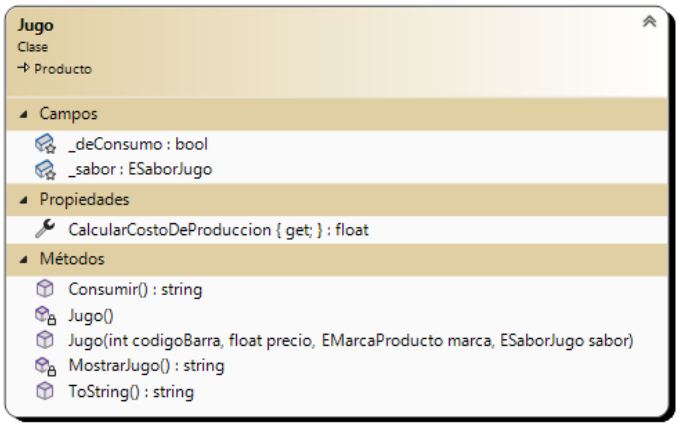
- Todos sus atributos son protegidos.
- Posee sólo un constructor de ***instancia***.
- La propiedad *Marca*, retornará el valor correspondiente del atributo *_marca*.
- La propiedad *Precio*, retornará el valor asociado al atributo *_precio*.
- La propiedad *CodigoDeBarra*, retornará el valor asociado al atributo *_codigoDeBarra*.
- El método privado de ***clase*** *MostrarProducto* retornará una cadena detallando los atributos de la clase.
- Sobrecarga de operadores:
  - **Igualdad** (Producto, Producto). Retornará *true*, si son del mismo tipo, 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.
  - **Explicito**. Retornará el código de barra del producto que recibe como parámetro.
  - **Implícito**. Retornará toda la información del producto. Reutilizar código.



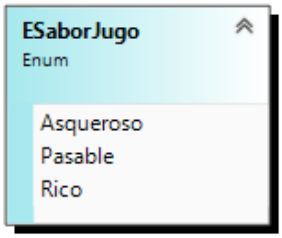
- El método *Equals* deberá indicar si dos objetos son del mismo tipo. Realizarlo en una línea de código.
- Contendrá una propiedad abstracta de sólo lectura llamada *CalcularCostoDeProduccion*. Retornará un *float*.
- Contendrá un método virtual llamado *Consumir* que retornará el *string "Parte de una mezcla."*.

También tendrá las siguientes clases derivadas de **Producto**:

Jugo:



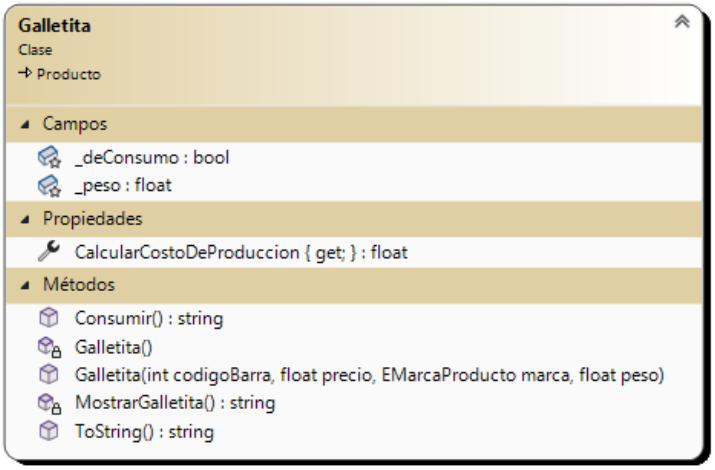
- Posee un atributo propio, que será inicializado por su único constructor de ***instancia***.
- El constructor de ***clase*** inicializará el atributo *_deConsumo* en *true*.
- El método privado de ***instancia*** *MostrarJugo*, retornará una cadena conteniendo la información completa del objeto. Reutilizar código.



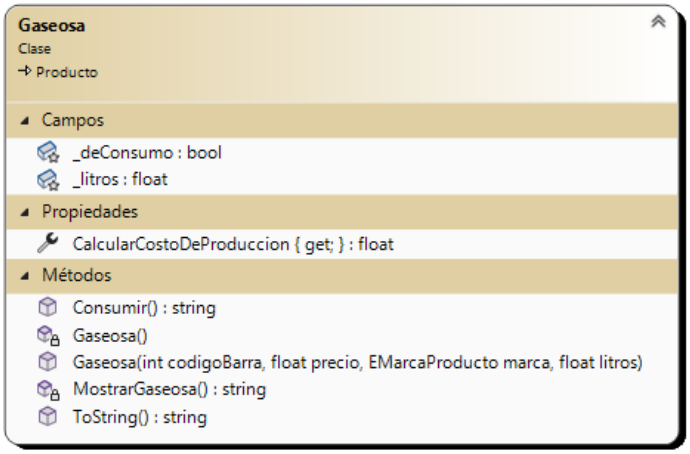
- El método *ToString* hará públicos los datos del producto.
- *CalcularCostoDeProduccion* será el 40% del precio final.
- *Consumir* retornará "*Bebible*".

Galletita:

- Posee un atributo propio, que será inicializado por su único constructor.
- El constructor de clase inicializará el atributo *_deConsumo* en *true*.
- El método privado de ***clase*** *MostrarGalletita*, retornará una cadena conteniendo la información completa del objeto recibido por parámetro.
- *CalcularCostoDeProduccion* será el 33% del precio final.
- El método *ToString* hará públicos los datos del producto.
- *Consumir* retornará "*Comestible*".

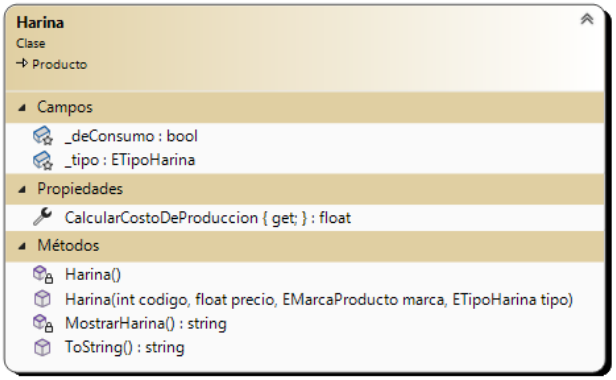


Gaseosa:



- Posee un único atributo propio, que será inicializado por su por una de las sobrecargas del constructor. Reutilizar código.
- El constructor de clase inicializará el atributo *_deConsumo* en *true*.
- El método privado de instancia *MostrarGaseosa*, retornará una cadena conteniendo la información completa del objeto.
- *CalcularCostoDeProduccion* será el 42% del precio final.
- El método *ToString* hará públicos los datos del producto.
- *Consumir* retornará "*Bebible*".

Harina:

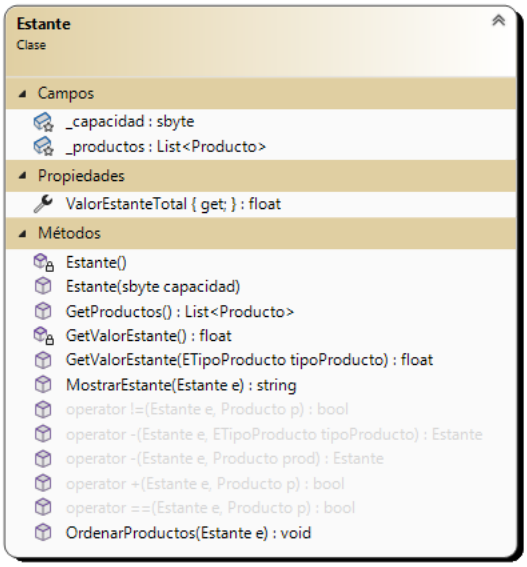


- Posee un único atributo propio, que será inicializado por su por una de las sobrecargas del constructor. Reutilizar código.
- El constructor de clase inicializará el atributo *_deConsumo* en *false*.
- El método privado de instancia *MostrarHarina*, retornará una cadena conteniendo la información completa del objeto.
- *CalcularCostoDeProducción* será el 60% del precio final.
- El método *ToString* hará públicos los datos del producto.

La última clase del proyecto será **Estante:**

- 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.
- 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.
- 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.

- 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;

- 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.
- Realizar un método llamado *GuardarEstante* de ***clase***. El método deberá guardar en un archivo de texto toda la información del estante y sus productos.
- Realizar un método llamado *SerializarEstante* de ***clase***. El método deberá guardar en un archivo XML toda la información del estante y sus productos.
- Realizar un método llamado *DeserializarEstante* de ***clase***. El método deberá leer el archivo XML con toda la información del estante y sus productos.

Agregar un método de ***clase*** que permita ordenar la lista de productos del estante a través del método *Sort* de dicha lista genérica a través de su Código de Barras.

Salida por consola

```
static void Main(string[] args)
{
    Estante estatenteUno = new Estante(4);
    Estante estanteDos = new Estante(3);
    Harina h1 = new Harina(102, 37.5f, EMarcaProducto.Favorita,
ETipoHarina.CuatroCeros);
    Harina h2 = new Harina(103, 40.25f, EMarcaProducto.Favorita,
ETipoHarina.Integral);
    Galletita g1 = new Galletita(113, 33.65f, EMarcaProducto.Pitusas, 250f);
    Galletita g2 = new Galletita(111, 56f, EMarcaProducto.Diversion, 500f);
    Jugo j1 = new Jugo(112, 25f,EMarcaProducto.Naranju, ESaborJugo.Pasable);
    Jugo j2 = new Jugo(333, 33f, EMarcaProducto.Swift,
ESaborJugo.Asqueroso);
    Gaseosa g = new Gaseosa(j2, 2250f);

    if (!(estatenteUno + h1))
    {
        Console.WriteLine("No se pudo agregar el producto al estante.");
```

```
    }
    if (!(estatenteUno + g1))
    {
        Console.WriteLine("No se pudo agregar el producto al estante.");
    }
    if (!(estatenteUno + g2))
    {
        Console.WriteLine("No se pudo agregar el producto al estante.");
    }
    if (!(estatenteUno + g1))
    {
        Console.WriteLine("No se pudo agregar el producto al estante.");
    }
    if (!(estatenteUno + j1))
    {
        Console.WriteLine("No se pudo agregar el producto al estante.");
    }
    if (!(estanteDos + h2))
    {
        Console.WriteLine("No se pudo agregar el producto al estante.");
    }
    if (!(estanteDos + j2))
    {
        Console.WriteLine("No se pudo agregar el producto al estante!!!");
    }
    if (!(estanteDos + g))
    {
        Console.WriteLine("No se pudo agregar el producto al estante.");
    }
    if (!(estanteDos + g1))
    {
        Console.WriteLine("No se pudo agregar el producto al estante!!!");
    }

    Console.WriteLine("\n******** INFORMACIÓN DE LOS ESTANTES ********");
    Console.WriteLine("- Valor total primer estante: ${0:##.##}",
estatenteUno.ValorEstanteTotal);
    Console.WriteLine("- Valor del primer estante sólo de Galletitas:
${0:##.##}",estatenteUno.GetValorEstante(ETipoProducto.Galletita));
    Console.WriteLine("\n=== Contenido primer estante ===\n{0}",
Estante.MostrarEstante(estatenteUno));
    Console.WriteLine("== Primer estante ordenado ===");
    estatenteUno.GetProductos().Sort(OrdenarProductos);
    Console.WriteLine(Estante.MostrarEstante(estatenteUno));

    estatenteUno = estatenteUno - ETipoProducto.Galletita;
    Console.WriteLine("=== Primer estante sin galletitas ===\n{0}",
Estante.MostrarEstante(estatenteUno));

    Console.WriteLine("=== Contenido segundo estante ===\n{0}",
Estante.MostrarEstante(estanteDos));
    estanteDos -= ETipoProducto.Todos;
    Console.WriteLine("=== Contenido segundo estante ===\n{0}",
Estante.MostrarEstante(estanteDos));
    Console.ReadLine();
    Console.Clear();

    string rutaApplicationData =
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    const string nombreArchivo = "listaProductos.txt";
    string rutaArchivo = Path.Combine(rutaApplicationData, nombreArchivo);
```

```
    Console.WriteLine("Guardando información del primer estante en un
archivo");
    Estante.GuardarEstante(estatenteUno, rutaArchivo);
    Console.ReadLine();

    const string nombreXML = "listaProductos.xml";
    rutaArchivo = Path.Combine(rutaApplicationData, nombreXML);

    Console.WriteLine("Serializando información del primer estante");
    Estante.SerializarEstante(estatenteUno, rutaArchivo);
    Console.ReadKey();

    Console.WriteLine("Serializando información del primer estante");
    Estante.DeserializarEstante(rutaArchivo);
    Console.ReadKey();
}
```