

---

# TAREA 3

---

TAREAS CAPACITACIÓN C#

📧 Ezequiel Remus  
ezequielremus@gmail.com

22 de marzo de 2023

## RESUMEN

### 1. Tarea Diagrama Objetos

#### 1.1. Solución

En la Figura 1 se ve una posible representación del sistema de clases del Enunciado anterior. Donde los Artículos e Instalaciones podrían tener otros atributos adicionales. En este caso, asumimos que quizás pueden existir más **artículos** del mismo (ejemplo, muchas pelotas iguales), por lo que se le podría asignar un atributo *cantidad*.

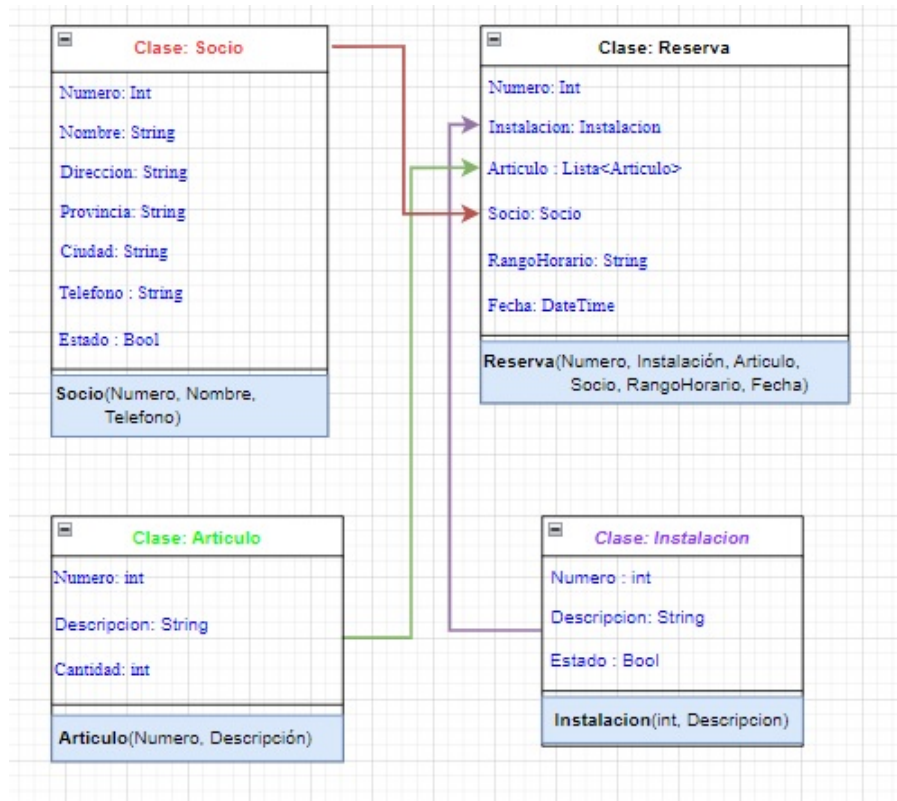


Figura 1: Diagrama de clases

Luego, podríamos suponer para las **instalaciones** que existían por ejemplo solo una pileta, o una sola cancha de fútbol por lo que se le asignaría a esta un estado para conocer si están o no en uso.

Estas no necesariamente están indicadas en el sistema, son cosas que se me ocurrieron a mí que podrían llegar a pasar

## 2. Tarea Practica Objetos

### 2.1. Crear una clase Coche con las siguientes propiedades: – Numero – Marca – Modelo – Año – Precio – Color

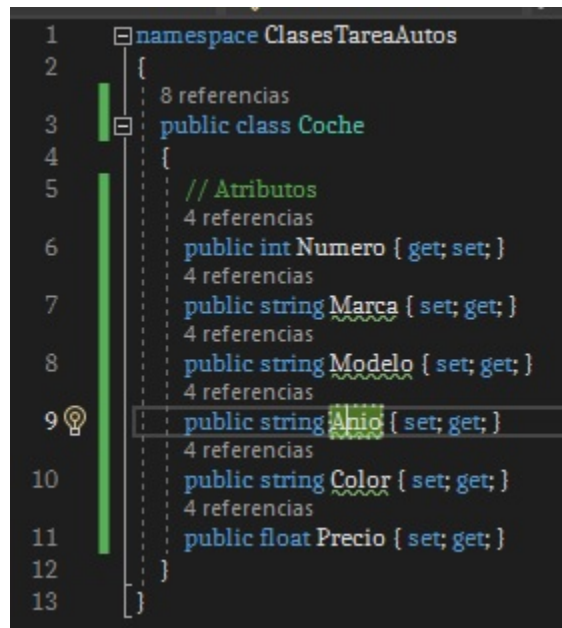


Figura 2: Diagrama de clases

### 2.2. Crear una clase Concesionario que gestione una serie de coches. Tendrá un List de objetos coches y además Numero, dirección, nombre y teléfono.

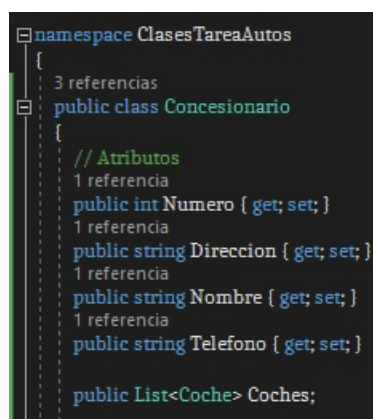


Figura 3: Concesionario. Atributos

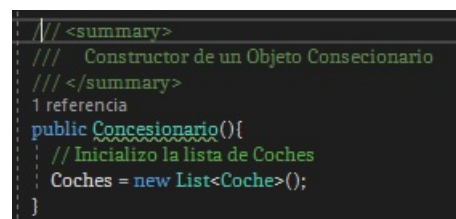


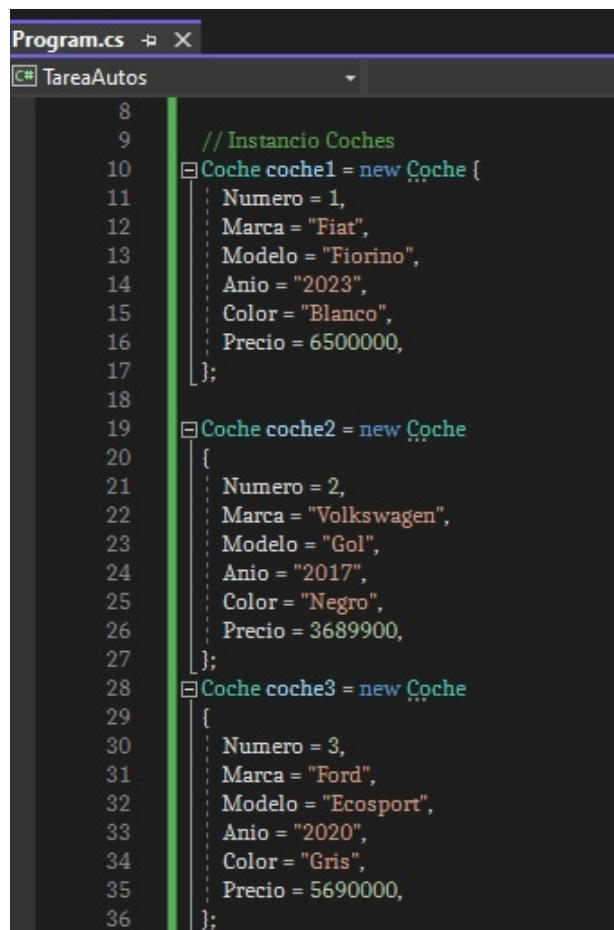
Figura 4: Constructor

### 2.3. Crear un Método MostrarStock(), que muestre todos los datos de los coches que hay en la concesionaria

```
/// <summary>
///  Muestra una lista de los coches de
///  un concesionario
/// </summary>
1 referencia
public void MostrarStock() {
    Console.WriteLine("-----\nLista de Coches\n-----\n");
    foreach (var coche in this.Coches)
    {
        Console.WriteLine(coche.Numero + " " + coche.Marca + " " + coche.Modelo + " " + coche.Color + " " + coche.Anio);
        Console.WriteLine("Precio: " + " $" + coche.Precio + " "
            + "\n\n");
    }
}
```

Figura 5: Metodo MostrarStock()

### 2.4. Instanciar 3 coches e instanciarlos con todos sus datos



```
Program.cs  X
C# TareaAutos
8
9 // Instancio Coches
10 Car coche1 = new Car {
11     Numero = 1,
12     Marca = "Fiat",
13     Modelo = "Fiorino",
14     Anio = "2023",
15     Color = "Blanco",
16     Precio = 6500000,
17 };
18
19 Car coche2 = new Car
20 {
21     Numero = 2,
22     Marca = "Volkswagen",
23     Modelo = "Gol",
24     Anio = "2017",
25     Color = "Negro",
26     Precio = 3689900,
27 };
28 Car coche3 = new Car
29 {
30     Numero = 3,
31     Marca = "Ford",
32     Modelo = "Ecosport",
33     Anio = "2020",
34     Color = "Gris",
35     Precio = 5690000,
36 };
```

Figura 6: Instancio tres coches

2.5. Instanciar un concesionario con todos sus datos y agregarle los 3 coches.

```
// Instancio Concesionario
Concesionario ConcesionarioUno = new Concesionario
{
    Numero = 1,
    Nombre = "Carone",
    Direccion = "Av. Maipu 3565",
    Telefono = "011 4794-0605",
};
ConcesionarioUno.Coches.Add(coche1);
ConcesionarioUno.Coches.Add(coche2);
ConcesionarioUno.Coches.Add(coche3);
```

Figura 7: Instancio tres coches

2.6. Mostrar por consola sus datos junto con el stock de coches que tiene.

```
// Utilizo la función para mostrar la lista de Autos del Concesionario
ConcesionarioUno.MostrarStock();
```

Figura 8: Utilizo la Función Necesaria

```
Consola de depuración de Microsoft Visual Studio

-----
Lista de Coches
-----

1 Fiat Fiorino Blanco 2023
Precio: 6500000

2 Volkswagen Gol Negro 2017
Precio: 3689900

3 Ford Ecosport Gris 2020
Precio: 5690000
```

Figura 9: Salida por consola

### 3. Tarea con el código de clase Código

#### 3.1. Agregar un atributo “Costo” al artículo

```
3 referencias
public class Artículo
{
    //defino mis atributos
    2 referencias
    public string Nombre { get; set; }
    2 referencias
    public int Numero { get; set; }
    0 referencias
    public string Categoria { get; set; }
    1 referencia
    public string Marca { get; set; }
    0 referencias
    public float MinimoStock { get; set; }
    0 referencias
    public string Proveedor { get; set; }
    5 referencias
    public float Costo { get; set; }
    0 referencias
    public float Precio { get; set; }
    public List<Categoria> Categorías;
}
```

Figura 10: Agrego Atributo Costo

#### 3.2. Agregar un método CalcularPrecio()

Agregar un método al artículo, CalcularPrecio() que devuelva un float con el precio de venta del artículo. El precio del artículo se calcula agregando un Margen de ganancia del 40 % al costo del Artículo. Por ejemplo si el costo el artículo es \$1000, su precio de venta sera \$1400.

```
/// <summary>
/// Calcula el precio del artículo
/// </summary>
/// <returns></returns>
0 referencias
public float CalcularPrecio() {
    return this.Costo + ((this.Costo * 40) / 100);
}
```

Figura 11: Código del Método

#### 3.3. Instanciar un artículo con todos sus atributos y mostrar por consola su precio

```
// Instancio Un Artículo
Articulo Harina = new Articulo {
    Numero = 1,
    Nombre = "Harina 0000",
    Marca = "Blanca Flor",
    MinimoStock = 5,
    Proveedor = "Distribuidora de Harinas S.R.L",
    Costo = 140,
};
Harina.Categorias.Add(CategoriaAlimento);
Harina.Precio = Harina.CalcularPrecio();
```

Figura 12: Instancio el articulo a mostrar

```
Console.WriteLine($"El precio del Artículo: {Harina.Nombre} es ${Harina.Precio} ");
```

Figura 13: Códigos para mostrar

C# Consola de depuración de Microsoft Visual Studio

El precio del Artículo: Harina 0000 es \$196

Figura 14: Salida por consola

- 3.4. Modificar el método “CalcularPrecio” del punto anterior para que si el precio de venta es mayor a \$10,000 se le reste \$1500

```
/// <summary>
/// Calcula el precio del articulo
/// </summary>
/// <returns></returns>
1 referencia
public float CalcularPrecio() {
    float Precio = this.Costo + ((this.Costo * 40) / 100);
    if (this.Costo > 10000)
    {
        return Precio - 1500;
    }
    else
    {
        return Precio;
    }
}
```

Figura 15: Modifico CalcularPrecio

### 3.5. Instanciar por los menos 2 artículos

Instanciar por los menos 2 artículos con todos sus atributos y mostrar por consola su precio, uno deberá superar los \$10,000 de precio de venta y otro no deberá superarlo. Mostrar por consola los datos del articulo incluyendo su precio.



```

// Instancio Un Artículo
Artículo Harina = new Artículo {
    Numero = 1,
    Nombre = "Harina 0000",
    Marca = "Blanca Flor",
    MinimoStock = 5,
    Proveedor = "Distribuidora de Harinas S.R.L",
    Costo = 140,
};
Harina.Categorías.Add(CategoríaAlimento);
Harina.Precio = Harina.CalcularPrecio();

// Instancio un segundo Artículo
Artículo Pantalon = new Artículo
{
    Numero = 2,
    Nombre = "Pantalon Jean",
    Marca = "Zara",
    MinimoStock = 2,
    Proveedor = "Inditex",
    Costo = 12000,
};
Pantalon.Categorías.Add(CategoríaIndumentaria);
Pantalon.Precio = Pantalon.CalcularPrecio();

```

Figura 16: Instancio Dos Artículos

```

2 referencias
public void MostrarArticulo() {
    Console.WriteLine($"{this.Numero} : {this.Nombre} {this.Marca}");
    Console.WriteLine($"{this.Minimo stock: {this.MinimoStock}");
    Console.WriteLine($"{this.Provee: {this.Proveedor}");
    Console.WriteLine($"{this.Precio: $ {this.Precio} \t \n");
}

```

Figura 17: Creo un método para Mostrar los artículos

```

// Muestro ambos artículos
Harina.MostrarArticulo();
Pantalon.MostrarArticulo();

```

Figura 18: Uso Metodo MostrarArticulo()

```

C# Consola de depuración de Microsoft Visual Studio

1 : Harina 0000 Blanca Flor
    Minimo stock: 5
    Provee: Distribuidora de Harinas S.R.L
-----
    Precio: $196
-----

2 : Pantalon Jean Zara
    Minimo stock: 2
    Provee: Inditex
-----
    Precio: $15300
-----

```

Figura 19: Salida por Consola

### 3.6. Hacer una modificación en la clase articulo

Hacer una modificación en la clase articulo (atributos y/o métodos) que me permita tener márgenes de ganancia distintos por cada articulo. De forma que pueda definir para el articulo su margen de ganancia y luego calcular su precio de venta. Deberá mantenerse la condición del punto anterior si el precio supera los \$10,000. Ejemplo: puedo definir el articulo Harina con un 30 % de margen de ganancia y el articulo Arroz con un 50 % de Margen de ganancia.

Para esto modifico los atributos del articulo agregando un **MargenGanancia**. Y ahora modificamos otra vez el método

```
namespace AppObjetos
{
    7 referencias
    public class Articulo
    {
        //defino mis atributos
        5 referencias
        public string Nombre { get; set; }
        5 referencias
        public int Numero { get; set; }
        0 referencias
        public string Categoria { get; set; }
        4 referencias
        public string Marca { get; set; }
        3 referencias
        public float MinimoStock { get; set; }
        3 referencias
        public string Proveedor { get; set; }
        5 referencias
        public float Costo { get; set; }
        3 referencias
        public float Precio { get; set; }
        1 referencia
        public float MargenGanancia { get; set; }
        public List<Categoria> Categorias;
    }
}
```

Figura 20: Agrego MargenGanancia al articulo

**CalcularPrecio** para que se calcule este segun el Margen de ganancia deseado. Esta sigue realizando el descuento segun el costo indicado.

```
/// <summary>
/// Calcula el precio del articulo
/// </summary>
/// <returns></returns>
2 referencias
public float CalcularPrecio() {
    float Precio = this.Costo + ((this.Costo * this.MargenGanancia) / 100);
    if (this.Costo > 10000)
    {
        return Precio - 1500;
    }
    else
    {
        return Precio;
    }
}
```

Figura 21: Modifico CalcularPrecio



### 3.7. Instanciamos Articulos y Mostramos

Instanciar por los menos 2 artículos con todos sus atributos y mostrar por consola su precio y margen de ganancia. Mostrar por consola los datos del artículo incluyendo su precio.

En este caso, solo debemos agregar el margen de ganancia a cada artículo. Luego, al calcular el precio ya tendrá ese

```
// Instancio Un Artículo
Articulo Harina = new Articulo {
    Numero = 1,
    Nombre = "Harina 0000",
    Marca = "Blanca Flor",
    MinimoStock = 5,
    Proveedor = "Distribuidora de Harinas S.R.L",
    Costo = 140,
    MargenGanancia = 50,
};
Harina.Categorias.Add(CategoriaAlimento);
Harina.Precio = Harina.CalcularPrecio();

// Instancio un segundo Artículo
Articulo Pantalon = new Articulo
{
    Numero = 2,
    Nombre = "Pantalon Jean",
    Marca = "Zara",
    MinimoStock = 2,
    Proveedor = "Inditex",
    Costo = 12000,
    MargenGanancia = 60,
};
Pantalon.Categorias.Add(CategoriaIndumentaria);
Pantalon.Precio = Pantalon.CalcularPrecio();
```

Figura 22: Instancia Agregando MargenGanancia

margen para poder calcular el precio del artículo.

Además, creamos un método **Gana()** que nos dice cuanto se gana por cada uno de estos artículos vendidos.

```
/// <summary>
/// La función se fija si ya se calculó el precio, sino
/// es así lo calcula y luego devuelve el margen de
/// ganancia en la moneda del costo.
/// </summary>
/// <returns></returns>
2 referencias
public float Gana() {
    if (this.Precio != null)
    {
        this.Precio = this.CalcularPrecio();
        return this.Precio - this.Costo;
    }
    else {
        return this.Precio - this.Costo;
    }
}
```

Figura 23: Método Gana

Mostramos por consola el artículo y por otro lado cuanto se gana con cada artículo.

```
// Muestro ambos articulos
Harina.MostrarArticulo();
Pantalon.MostrarArticulo();

Console.WriteLine($" * Con cada Articulo {Harina.Nombre} se gana ${Harina.Gana()}");
Console.WriteLine($" * Con cada Articulo {Pantalon.Nombre} se gana ${Pantalon.Gana()}");
```

Figura 24: Código para mostrar

```
C:\Users\Hp\Desktop\Codigo\AppObjetos\AppObjetos\bin\Debug\net6.0\AppO
1 : Harina 0000 Blanca Flor
    Minimo stock: 5
    Provee: Distribuidora de Harinas S.R.L
-----
    Precio: $210
-----
2 : Pantalon Jean Zara
    Minimo stock: 2
    Provee: Inditex
-----
    Precio: $17700
-----
* Con cada Articulo Harina 0000 se gana $70
* Con cada Articulo Pantalon Jean se gana $5700
```

Figura 25: Muestra

### 3.8. Agregar como atributo una lista de ArticulosPermitidos a la clase Deposito.

Agregar como atributo una lista de ArticulosPermitidos.<sup>a</sup> la clase Deposito. (No olvidar inicializar la lista en el constructor)

```
public List<Articulo> ArticulosPermitidos;

/// <summary>
/// Constructor del Deposito
/// </summary>
1 referencia
public Deposito()
{
    //inicializo la lista de categorias
    ArticulosPermitidos = new List<Articulo>();
}
```

Figura 26: Agrego lista de articulos Permitidos + Constructor

Instanciar 3 Artículos con sus datos

```
// Instancio Un Articulo
Articulo Arroz = new Articulo
{
    Numero = 3,
    Nombre = "Arroz Gallo Oro",
    Marca = "Gallo",
    MinimoStock = 3,
    Proveedor = "Distribuidora Mayorista Dotta",
    Costo = 300,
    MargenGanancia = 50,
};
Harina.Categorias.Add(CategoriaAlimento);
Harina.Precio = Harina.CalcularPrecio();
```

Figura 27: Instancio Un nuevo Articulo

Creo Funciones para mostrar Depositos

```
/// <summary>
/// Devuelve un string con un mensaje que indica las variables principales
/// de un deposito
/// </summary>
/// <returns></returns>
1 referencia
public string MostrarVariablesPrincipales()
{
    return "El numero del deposito es " + this.Numero + " Nombre: " + this.Nombre;
}

/// <summary>
/// Entrega un mensaje con los Articulos permitidos
/// </summary>
/// <returns></returns>
1 referencia
public void MostrarArticulosPermitidos() {
    Console.WriteLine("\t*****\n"
        + "\t Articulos Permitidos\n\t*****\n");
    foreach (var articulo in this.ArticulosPermitidos)
    {
        Console.WriteLine($"{articulo.Numero} : {articulo.Nombre} \n");
    }
}

/// <summary>
/// Muestra los datos Completos de un deposito
/// </summary>
1 referencia
public void MostrarDeposito() {
    Console.WriteLine(MostrarVariablesPrincipales());
    MostrarArticulosPermitidos();
}
```

Figura 28: Metodos que utilizo para mostrar los depositos con sus articulos

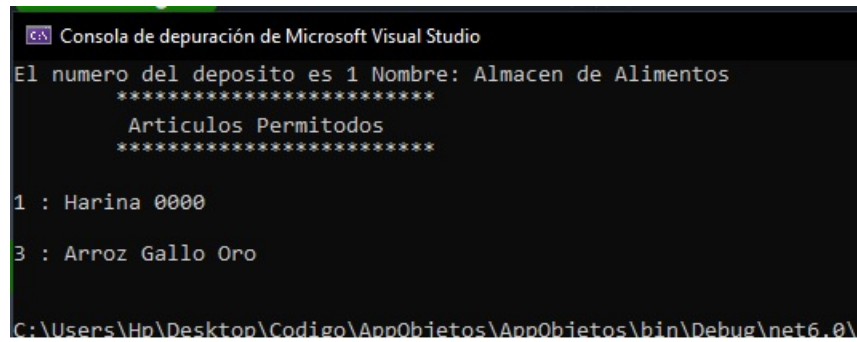
Instanciar 1 Deposito con sus datos y agregar a la lista de permitidos 2 de estos artículos.

```
// Inicializo un Deposito
Deposito DepositoUno = new Deposito
{
    Numero = 1,
    Nombre = "Almacen de Alimentos",
    Capacidad = 100,
    Direccion = "Perito Moreno 845",
};
DepositoUno.ArticulosPermitidos.Add(Harina);
DepositoUno.ArticulosPermitidos.Add(Arroz);

// Muestro los datos principales y los articulos permitidos
DepositoUno.MostrarDeposito();
```

Figura 29: Metodos que utilizo para mostrar los depositos con sus articulos

Mostrar por pantallas los datos del Deposito mas el numero y nombre de los artículos permitidos.



```
Consola de depuración de Microsoft Visual Studio
El numero del deposito es 1 Nombre: Almacen de Alimentos
*****
    Articulos Permitidos
*****

1 : Harina 0000
3 : Arroz Gallo Oro

C:\Users\Ho\Desktop\Codigo\AppObjetos\AppObjetos\bin\Debug\net6.0\
```

Figura 30: Salida por consola

### 3.9. Crear un método en la clase deposito ValidarArticulo

Crear un método en la clase deposito “ValidarArticulo” que reciba como parámetro un “articulo” y devuelva un booleano si el articulo se acepta o no en el deposito. Deberá validarse si el articulo recibido esta en la lista de “ArticulosPermitidos” agregado en el punto anterior

Para la solución Cree el siguiente metodo en la clase Deposito

```
3 referencias
public bool ValidarArticulo(Articulo articulo) {
    foreach (var art in this.ArticulosPermitidos)
    {
        if (art.Nombre == articulo.Nombre)
        {
            return true;
        }
    }
    return false;
}
```

Figura 31: Metodos ValidarArticulo

Instanciar 3 Artículos con sus datos Instanciar 1 Deposito con sus datos y agregar a la lista de permitidos 2 de estos artículos. Usar el método de validación creado y llamarlo por cada articulo mostrando por consola si el articulo se permite o no en el deposito

Con los articulos ya inicializados anteriormente y el deposito creado en el ejercicio anterior aplico la validación

```
// Inicializo un Deposito
Deposito DepositoUno = new Deposito
{
    Numero = 1,
    Nombre = "Almacen de Alimentos",
    Capacidad = 100,
    Direccion = "Perito Moreno 845",
};

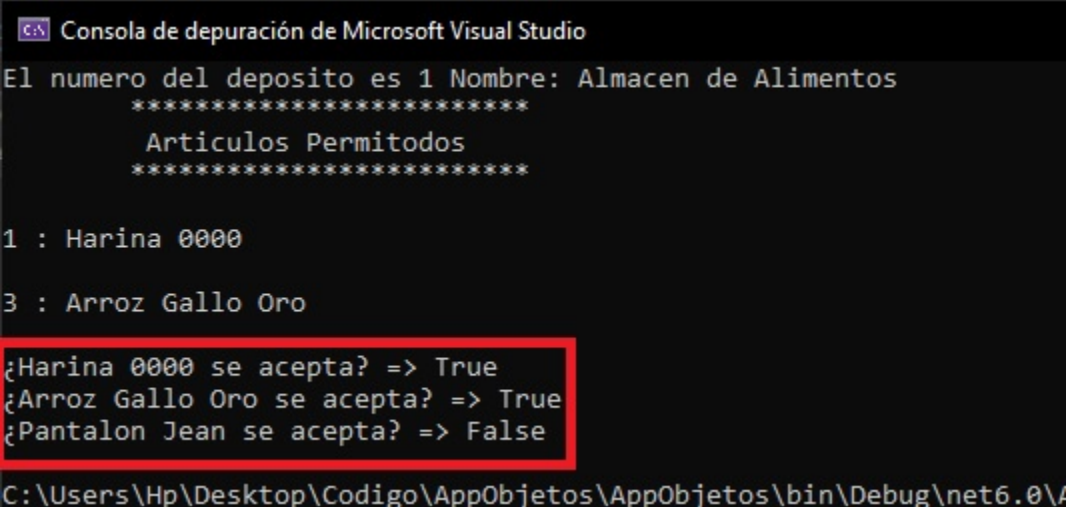
// Agrego ArticulosPermitidos por el deposito
DepositoUno.ArticulosPermitidos.Add(Harina);
DepositoUno.ArticulosPermitidos.Add(Arroz);

// Muestro los datos principales y los articulos permitidos
DepositoUno.MostrarDeposito();

// Valido articulos que pueden ingresar al DepositoUno
Console.WriteLine($"{Harina.Nombre} se acepta? => {DepositoUno.ValidarArticulo(Harina)}");
Console.WriteLine($"{Arroz.Nombre} se acepta? => {DepositoUno.ValidarArticulo(Arroz)}");
Console.WriteLine($"{Pantalon.Nombre} se acepta? => {DepositoUno.ValidarArticulo(Pantalon)}");
```

Figura 32:Codigo a ejecutar

Luego, vemos lo siguiente en la consola:



```

Consola de depuración de Microsoft Visual Studio
El numero del deposito es 1 Nombre: Almacen de Alimentos
*****
Articulos Permitodos
*****

1 : Harina 0000

3 : Arroz Gallo Oro

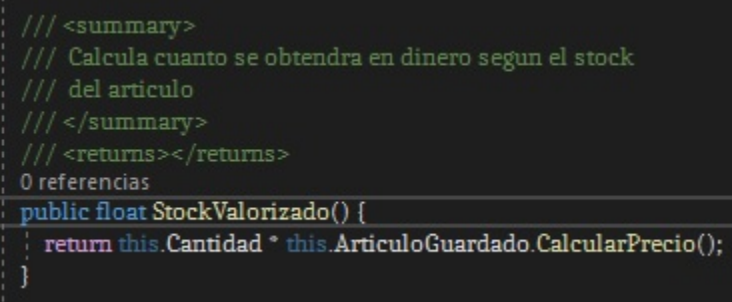
¿Harina 0000 se acepta? => True
¿Arroz Gallo Oro se acepta? => True
¿Pantalon Jean se acepta? => False

C:\Users\Hp\Desktop\Codigo\AppObjetos\AppObjetos\bin\Debug\net6.0\A
  
```

Figura 33: Salida por consola

### 3.10. Agregar un metodo en la Clase stock “StockValorizado()”

Agregar un método en la Clase stock “StockValorizado()”, que devuelva la cantidad de stock por el precio del articulo(Reutilizar el método “CalcularPrecio()” de la clase Articulo). Ejemplo si tengo un Stock del articulo Harina de 10 unidades y el costo es \$500, deberá devolver Stock Valorizado \$5000



```

/// <summary>
/// Calcula cuanto se obtendra en dinero segun el stock
/// del articulo
/// </summary>
/// <returns></returns>
0 referencias
public float StockValorizado() {
    return this.Cantidad * this.ArticuloGuardado.CalcularPrecio();
}
  
```

Figura 34: Metodo ValorizarStock

Notemos que la cuenta que realiza es

$$StockValorizado = CantidadStock \cdot PrecioArticulo$$

Instanciar un articulo con sus datos Instanciar un deposito con sus datos Instanciar un Stock, agregarle articulo, deposito y una cantidad

En este caso, reciclamos el deposito y los artículos anteriores, en particular el articulo Harina



```
// Instancio Un Artículo
Articulo Harina = new Articulo {
    Numero = 1,
    Nombre = "Harina 0000",
    Marca = "Blanca Flor",
    MínimoStock = 5,
    Proveedor = "Distribuidora de Harinas S.R.L",
    Costo = 140,
    MargenGanancia = 50,
};
Harina.Categorias.Add(CategoriaAlimento);
Harina.Precio = Harina.CalcularPrecio();

// Inicializo un Deposito
Deposito DepositoUno = new Deposito
{
    Numero = 1,
    Nombre = "Almacen de Alimentos",
    Capacidad = 100,
    Direccion = "Perito Moreno 845",
};
// Agrego Articulos Permitidos por el deposito
DepositoUno.ArticulosPermitidos.Add(Harina);
DepositoUno.ArticulosPermitidos.Add(Arroz);

Stock miStock = new Stock {
    ArticuloGuardado = Harina,
    DepositoDondeEstaGuardado = DepositoUno,
    Cantidad = 10
};

Console.WriteLine($"El stockValorizado de {miStock.ArticuloGuardado.Nombre} es: {miStock.StockValorizado()}");
```

Figura 35: Código a ejecutar

Mostrar por consola el Stock valorizado.

```
C:\> Consola de depuración de Microsoft Visual Studio
El stockValorizado de Harina 0000 es: 2100
C:\Users\Hp\Desktop\Codigo\AppObjetos\AppObjetos\
...
Para cerrar automáticamente la consola cuando se
```

Figura 36: Salida Por Consola

### 3.11. En la clase Stock agregar un método “AgregarArticulo()”

En la clase Stock agregar un método “AgregarArticulo()” que reciba como parámetro el artículo a agregar, valide si ese artículo está en la lista de “Artículos Permitidos” del depósito donde se está agregando stock. En caso de estar en la lista de artículos permitidos, deberá asignarlo al atributo Artículo de la clase stock y devolver el mensaje “Artículo Agregado” y en caso de no estar en los permitidos devolver el mensaje “Artículo No Permitido”.

```

    /// <summary>
    ///   Agrega un artículo teniendo en cuenta si este puede
    ///   ser despensado en el depósito asignado por stock
    /// </summary>
    /// <param name="articulo"></param>
    /// <returns></returns>
    3 referencias
    public string AgregarArticulo(Articulo articulo) {

        if (this.DepositoDondeEstaGuardado.ValidarArticulo(articulo) != false)
        {
            this.ArticuloGuardado = articulo;
            return "Artículo Agregado";
        }
        else {
            return "Artículo No Permitido";
        }
    }

```

Figura 37: Código del Método

Usar los 3 artículos del punto anterior y el depósito Instanciar una clase Stock, asignar el depósito y luego llamar al método “AgregarArticulo()”, pasándoles como parámetro alguno de los artículos generados que se permita agregar en el depósito. Instanciar otra clase Stock, asignar el depósito y luego llamar al método “AgregarArticulo()”, pasándoles como parámetro alguno de los artículos generados que NO se permita agregar en el depósito.

```

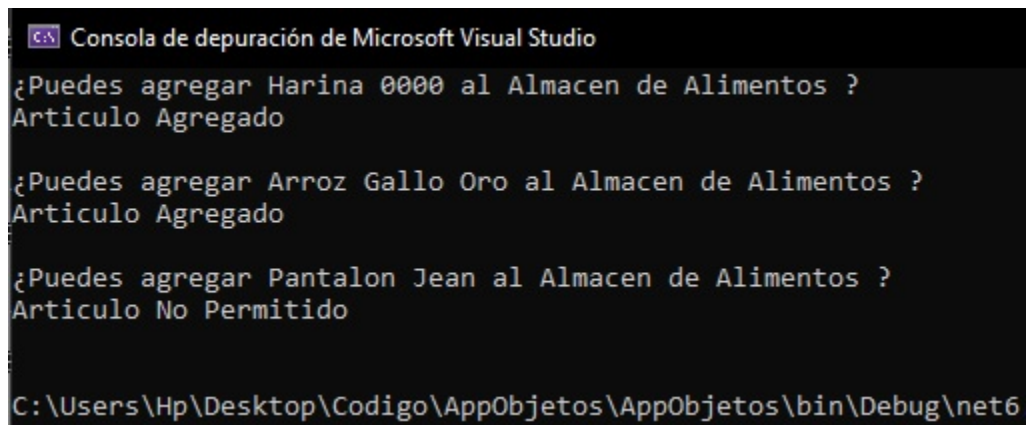
Stock OtroStock = new Stock
{
    DepositoDondeEstaGuardado = DepositoUno,
    Cantidad = 30,
};

// Intento Agregar los Artículos a OtroStock
Console.WriteLine($"¿Puedes agregar {Harina.Nombre} al {DepositoUno.Nombre} ?");
Console.WriteLine(OtroStock.AgregarArticulo(Harina) + "\n");
Console.WriteLine($"¿Puedes agregar {Arroz.Nombre} al {DepositoUno.Nombre} ?");
Console.WriteLine(OtroStock.AgregarArticulo(Arroz) + "\n");
Console.WriteLine($"¿Puedes agregar {Pantalon.Nombre} al {DepositoUno.Nombre} ?");
Console.WriteLine(OtroStock.AgregarArticulo(Pantalon) + "\n");

```

Figura 38: Instancio OtroStock con el depositoUno para realizar las pruebas

*Mostrar por consola los mensajes de si se agrego el articulo o no. Como se puede ver no se puede agregar el Articulo*



```
Consola de depuración de Microsoft Visual Studio
¿Puedes agregar Harina 0000 al Almacen de Alimentos ?
Articulo Agregado
¿Puedes agregar Arroz Gallo Oro al Almacen de Alimentos ?
Articulo Agregado
¿Puedes agregar Pantalon Jean al Almacen de Alimentos ?
Articulo No Permitido
C:\Users\Hp\Desktop\Codigo\AppObjetos\AppObjetos\bin\Debug\net6
```

Figura 39: Salida Por Consola

Pantalon, pues el DepositoUno solo tiene asignados Harina 0000 y Arroz Gallo.

## Referencias