Obligatorio Programación 2

Docente Luis Dentone

Grupo N2B

Cecilia Belon

256569

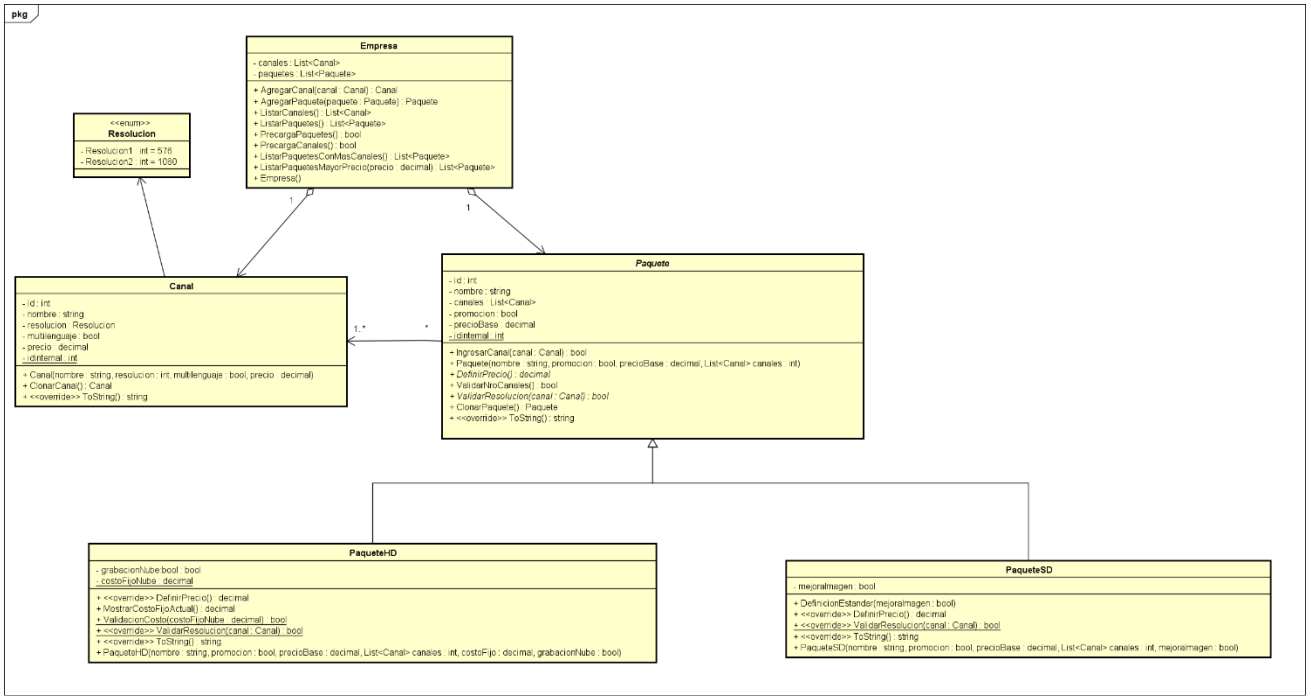
Sebastián Piazza

250224

Jose Sanchez

261466

1. Diagrama completo del Dominio del problema:



2. Tabla con la información de los datos precargados

   Tabla: Canales

   Canal(Id, Nombre, Resolución, Multilenguaje, precio);

id	nombre	resolucion	multilenjuage	precio
1	canal1	576	VERDADERO	180
2	canal2	576	VERDADERO	190
3	canal3	1080	VERDADERO	100
4	canal4	1080	VERDADERO	120
5	canal5	576	VERDADERO	130
6	canal6	576	VERDADERO	145
7	canal7	1080	VERDADERO	150
8	canal8	1080	VERDADERO	155
9	canal9	576	VERDADERO	160
10	canal10	576	VERDADERO	165
11	canal11	1080	VERDADERO	170
12	canal12	1080	VERDADERO	180
13	canal13	576	VERDADERO	180
14	canal14	1080	VERDADERO	181

15	canal15	1080	VERDADERO	182
16	canal16	576	VERDADERO	183
17	canal17	576	VERDADERO	184
18	canal18	1080	VERDADERO	185
19	canal19	1080	VERDADERO	186
20	canal20	576	VERDADERO	180
21	canal21	576	VERDADERO	180
22	canal22	1080	VERDADERO	1950
23	canal23	1080	VERDADERO	196
24	canal24	576	VERDADERO	112
25	canal25	576	VERDADERO	113
26	canal26	1080	VERDADERO	114
27	canal27	1080	VERDADERO	180
28	canal28	576	VERDADERO	116
29	canal29	576	VERDADERO	180
30	canal30	1080	VERDADERO	180
31	canal31	1080	VERDADERO	117
32	canal32	576	VERDADERO	118
33	canal33	576	VERDADERO	119
34	canal34	1080	VERDADERO	132
35	canal35	1080	VERDADERO	133
36	canal36	576	VERDADERO	134
37	canal37	576	VERDADERO	180
38	canal38	1080	VERDADERO	180
39	canal39	1080	VERDADERO	180
40	canal40	576	VERDADERO	180
41	canal41	576	VERDADERO	180
42	canal42	1080	VERDADERO	180
43	canal43	1080	VERDADERO	180
44	canal44	576	VERDADERO	180
45	canal45	576	VERDADERO	135
46	canal46	1080	VERDADERO	136
47	canal47	1080	VERDADERO	137
48	canal48	576	VERDADERO	188
49	canal49	576	VERDADERO	139

50	canal50	576	VERDADERO	184

Paquete HD:

PaqueteHD(Id, Nombre, Promoción, PrecioBase, GrabacionNube);

id	nombre	promocion	precioBase	grabacionNube
1	paqueteHD1	VERDADERO	580	VERDADERO
2	paqueteHD2	VERDADERO	523	VERDADERO
3	paqueteHD3	FALSO	623	VERDADERO
4	paqueteHD4	FALSO	871	VERDADERO
5	paqueteHD5	VERDADERO	174	VERDADERO
6	paqueteHD6	VERDADERO	823	VERDADERO
7	paqueteHD7	FALSO	284	VERDADERO
8	paqueteHD8	FALSO	937	VERDADERO
9	paqueteHD9	VERDADERO	127	VERDADERO
10	paqueteHD10	VERDADERO	264	VERDADERO

Paquete SD:

PaqueteSD(Id, Nombre, Promoción, PrecioBase, GrabacionNube);

id	nombre	promocion	precioBase	mejoraImagen
11	paqueteSD1	VERDADERO	385	VERDADERO
12	paqueteSD2	VERDADERO	396	VERDADERO
13	paqueteSD3	FALSO	295	VERDADERO
14	paqueteSD4	FALSO	285	VERDADERO
15	paqueteSD5	VERDADERO	184	VERDADERO
16	paqueteSD6	VERDADERO	195	VERDADERO
17	paqueteSD7	FALSO	186	VERDADERO
18	paqueteSD8	FALSO	275	VERDADERO
19	paqueteSD9	VERDADERO	275	VERDADERO
20	paqueteSD10	VERDADERO	195	VERDADERO

Tabla de relación “Canal pertenece a Paquete”:

CanalPaquete(IdPaquete, IdCanal);

id (Paquete)	id (canal)
1	3
1	4
1	7
1	8
11	1
11	2

3. Código fuente COMENTADO de toda la aplicación.

Solamente se incluirá el código fuente de las clases del dominio. Se deberán incluir los comentarios que considere relevantes.

- **Canal.s**

```
stem;

e Dominio

<summary>
Representa un Canal en el sistema
</summary>
ic class Canal

#region atributos
    private static int internalID=0;
    protected int Id;
    public string Nombre { get; set; }
    public Resolucion Resolucion { get; set; }
    public bool MultiLenguaje { get; set; }
    public decimal Precio { get; set; }
#endregion atributos

#region Constructores
public Canal(string nombre, Resolucion resolucion, bool multilenguaje, decimal precio)
{
    Id = ++internalID;
    Nombre = nombre;
    Resolucion = resolucion;
    MultiLenguaje = multilenguaje;
    Precio = precio;
}

public Canal(int id, string nombre, Resolucion resolucion, bool multilenguaje, decimal precio)
{
```

```
    Id = ++internalID;
    Nombre = nombre;
    Resolucion = resolucion;
    MultiLenguaje = multilenguaje;
    Precio = precio;
}
#endregion

#region metodos
public Canal ClonarCanal()
{
    Canal canalAux = new Canal(Id, Nombre, Resolucion, MultiLenguaje, Precio);
    return canalAux;
}

public static bool ValidarPrecio(decimal precio)
{
    return precio > 0;
}

public static bool ValidarNombre(string nombre)
{
    return nombre.Length > 3;
}

public override string ToString()
{
    return $"Nombre {Nombre} | Resolucion {(int)Resolucion} | MultiLenguaje: {MultiLenguaje} |
{Precio}";
}

internal int GetId()
{
    return Id;
}
#endregion


ic enum Resolucion : int

BAJA = 576,
ALTA = 1080
```

- **Paquete.cs**

```
using System;
using System.Collections.Generic;

namespace Dominio
{
    /// <summary>
    /// Representa un paquete con una lista de canales en el sistema
    /// </summary>
```

```
    public abstract class Paquete
    {

        #region atributos
        //id autogenerado
        private static int internalID=0;
        protected int Id;
        public string Nombre { get; set; }

        protected TipoPaquete tipoPaquete;


        private List<Canal> canales;
        public List<Canal> Canales { get { return canales; } }

        public bool Promocion { get; set; }
        public decimal PrecioBase { get; set; }
        #endregion

        #region constructores
        public Paquete(string nombre, bool promocion, decimal
precioBase,List<Canal> canales)
        {
            this.canales = canales;
            Id = ++internalID;
            Nombre = nombre;
            Promocion = promocion;
            PrecioBase = precioBase;
        }

        public Paquete(string nombre, bool promocion, decimal precioBase,
List<Canal> canales, int id)
        {
            this.canales = canales;
            Id = id;
            Nombre = nombre;
            Promocion = promocion;
            PrecioBase = precioBase;
        }
        #endregion

        #region metodos
        public bool IngresarCanal(Canal c)
        {
            canales.Add(c);
            return true;
        }

        internal int CantCanales()
        {
            return canales.Count;
        }

        public int GetId()
        {
            return Id;
        }

        public abstract Paquete ClonarPaquete();

        public virtual decimal DefinirPrecio()
        {
```

```
            decimal result=PrecioBase;
            foreach (Canal c in Canales)
            {
                result += c.Precio;
            }
            return result;
        }

        public bool ValidarCantCanales()
        {
            return canales.Count > 0;
        }

        public override string ToString()
        {
            return $"Nombre : {Nombre} | Promocion : {Promocion} | Precio Base :
{PrecioBase}";
        }
        #endregion

    }

    public enum TipoPaquete
    {
        SD =1,
        HD =2
    }
}
```

- **PaqueteHD.cs**

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Dominio
{
    /// <summary>
    /// Representa un Paquete HD (alta definicion) en el sistema
    /// </summary>
    public class PaqueteHD : Paquete
    {

        #region atributos
        //atributos y propiedades EN UNA LINEA SOLA
        public bool GrabacionNube { get; set; }
        public static decimal CostoFijo { get; set; }
        #endregion

        #region constructores
        public PaqueteHD(string nombre, bool promocion, decimal
precioBase,List<Canal> canales, bool grabacionNube) : base(nombre, promocion,
precioBase,canales)
        {
            tipoPaquete = TipoPaquete.HD;
            GrabacionNube = grabacionNube;
```

```
        }

        public PaqueteHD(string nombre, bool promocion, decimal precioBase,
List<Canal> canales, int id, bool grabacionNube) : base(nombre, promocion,
precioBase, canales, id)
        {
            tipoPaquete = TipoPaquete.HD;
            GrabacionNube = grabacionNube;
        }
        #endregion

        #region metodos
        public override Paquete ClonarPaquete()
        {
            return new PaqueteHD(Nombre, Promocion, PrecioBase, Canales, Id,
GrabacionNube);
        }

       public override decimal DefinirPrecio()
        {
            decimal result = base.DefinirPrecio();
            if (GrabacionNube == true && Promocion == false)
            {
                return result + CostoFijo;

            }
            if (GrabacionNube == true && Promocion == true)
            {
                return (result + CostoFijo) / 2;
            }
            return result ;
        }

        public static bool ValidarPrecioNube(decimal precioACambiar)
        {
            return precioACambiar > 0;
        }
        #endregion
    }
}
```

PaqueteSD.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Dominio
{
    /// <summary>
    /// Representa un Paquete SD ( definicion standart) en el sistema
    /// </summary>
    public class PaqueteSD : Paquete
    {
        #region atributos
        public bool MejoraImagen { get; set; }
        #endregion

        #region constructores
```

```
        public PaqueteSD(string nombre, bool promocion, decimal
precioBase,List<Canal> canales, bool mejoraimagen)
            : base(nombre, promocion, precioBase, canales)
        {
            tipoPaquete = TipoPaquete.SD;
            MejoraImagen = mejoraimagen;
        }
        public PaqueteSD(string nombre, bool promocion, decimal precioBase,
List<Canal> canales, int id, bool mejoraImagen)
             : base(nombre, promocion, precioBase, canales,id)
        {
            tipoPaquete = TipoPaquete.SD;
            MejoraImagen = mejoraImagen;
        }
        #endregion

        #region metodos

        /// <summary>
        /// Clonar un paquete sd a partir de los datos originales
        /// </summary>
        /// <returns>nueva instancia de paquetesd</returns>
        public override Paquete ClonarPaquete()
        {
            return new
PaqueteSD(Nombre,Promocion,PrecioBase,Canales,Id,MejoraImagen);
        }

        public override decimal DefinirPrecio()
        {
            decimal result = base.DefinirPrecio();

            //20% mas si tiene mejoramiento de imagen
            if (MejoraImagen)
            {
                result = ((decimal)1.2) * result;
            }
            //15% descuento si esta en promocion
            if (Promocion)
            {
                result = result - result * (decimal)0.15;
            }

            return result;
        }

        #endregion
    }
}
```

Empresa.cs

```
using System;
using System.Collections.Generic;

namespace Dominio
{


    public class Empresa
    {
        #region atributos
        public int Id { get; set; }
        public string Nombre { get; set; }
        public List<Paquete> Paquetes { get; } = new List<Paquete>();
        public List<Canal> Canales { get; } = new List<Canal>();
        #endregion

        #region constructores
        public Empresa()
        {
            PrecargaPaquetes();
            PrecargaCanales();
            PrecargaCanalesAPaquetes();
        }
        #endregion

        #region metodos

        /// <summary>
        /// precargar los paquetes con distintos datos
```

```
/// </summary>
public void PrecargaPaquetes()
{
    PaqueteHD.CostoFijo = 566;
    AgregarPaqueteHD("paqueteHD1", true, 580, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD2", true, 523, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD3", false, 623, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD4", false, 871, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD5", true, 174, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD6", true, 823, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD7", false, 284, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD8", false, 937, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD9", true, 127, true, new List<Canal>());
    AgregarPaqueteHD("paqueteHD10", true, 264, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD1", true, 385, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD2", true, 396, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD3", false, 295, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD4", false, 285, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD5", true, 184, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD6", true, 195, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD7", false, 186, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD8", false, 275, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD9", true, 275, true, new List<Canal>());
    AgregarPaqueteSD("paqueteSD10", true, 195, true, new List<Canal>());
}

/// <summary>
/// precarga los canales con distintos datos
/// </summary>
public void PrecargaCanales()
{
```

```
AgregarCanal("canal1", Resolucion.BAJA, true, 180);
AgregarCanal("canal2", Resolucion.BAJA, true, 190);
AgregarCanal("canal3", Resolucion.ALTA, true, 100);
AgregarCanal("canal4", Resolucion.ALTA, true, 120);
AgregarCanal("canal5", Resolucion.BAJA, true, 130);
AgregarCanal("canal6", Resolucion.BAJA, true, 145);
AgregarCanal("canal7", Resolucion.ALTA, true, 150);
AgregarCanal("canal8", Resolucion.ALTA, true, 155);
AgregarCanal("canal9", Resolucion.BAJA, true, 160);
AgregarCanal("canal10", Resolucion.BAJA, true, 165);
AgregarCanal("canal11", Resolucion.ALTA, true, 170);
AgregarCanal("canal12", Resolucion.ALTA, true, 180);
AgregarCanal("canal13", Resolucion.BAJA, true, 180);
AgregarCanal("canal14", Resolucion.ALTA, true, 181);
AgregarCanal("canal15", Resolucion.ALTA, true, 182);
AgregarCanal("canal16", Resolucion.BAJA, true, 183);
AgregarCanal("canal17", Resolucion.BAJA, true, 184);
AgregarCanal("canal18", Resolucion.ALTA, true, 185);
AgregarCanal("canal19", Resolucion.ALTA, true, 186);
AgregarCanal("canal20", Resolucion.BAJA, true, 180);
AgregarCanal("canal21", Resolucion.BAJA, true, 180);
AgregarCanal("canal22", Resolucion.ALTA, true, 1950);
AgregarCanal("canal23", Resolucion.ALTA, true, 196);
AgregarCanal("canal24", Resolucion.BAJA, true, 112);
AgregarCanal("canal25", Resolucion.BAJA, true, 113);
AgregarCanal("canal26", Resolucion.ALTA, true, 114);
AgregarCanal("canal27", Resolucion.ALTA, true, 180);
AgregarCanal("canal28", Resolucion.BAJA, true, 116);
AgregarCanal("canal29", Resolucion.BAJA, true, 180);
AgregarCanal("canal30", Resolucion.ALTA, true, 180);
AgregarCanal("canal31", Resolucion.ALTA, true, 117);
```

```
    AgregarCanal("canal32", Resolucion.BAJA, true, 118);
    AgregarCanal("canal33", Resolucion.BAJA, true, 119);
    AgregarCanal("canal34", Resolucion.ALTA, true, 132);
    AgregarCanal("canal35", Resolucion.ALTA, true, 133);
    AgregarCanal("canal36", Resolucion.BAJA, true, 134);
    AgregarCanal("canal37", Resolucion.BAJA, true, 180);
    AgregarCanal("canal38", Resolucion.ALTA, true, 180);
    AgregarCanal("canal39", Resolucion.ALTA, true, 180);
    AgregarCanal("canal40", Resolucion.BAJA, true, 180);
    AgregarCanal("canal41", Resolucion.BAJA, true, 180);
    AgregarCanal("canal42", Resolucion.ALTA, true, 180);
    AgregarCanal("canal43", Resolucion.ALTA, true, 180);
    AgregarCanal("canal44", Resolucion.BAJA, true, 180);
    AgregarCanal("canal45", Resolucion.BAJA, true, 135);
    AgregarCanal("canal46", Resolucion.ALTA, true, 136);
    AgregarCanal("canal47", Resolucion.ALTA, true, 137);
    AgregarCanal("canal48", Resolucion.BAJA, true, 188);
    AgregarCanal("canal49", Resolucion.BAJA, true, 139);
    AgregarCanal("canal50", Resolucion.BAJA, true, 184);
}

/// <summary>
/// asociar canales existentes a paquetes
/// </summary>
/// <returns></returns>
public string PrecargaCanalesAPaquetes()
{
    string errores = "";
    Paquete auxPaq1 = BuscarPaquete("paqueteHD1");
    auxPaq1.IngresarCanal(BuscarCanal("canal3"));
    auxPaq1.IngresarCanal(BuscarCanal("canal4"));
```

```
            auxPaq1.IngresarCanal(BuscarCanal("canal7"));
            auxPaq1.IngresarCanal(BuscarCanal("canal8"));
            if (!auxPaq1.IngresarCanal(BuscarCanal("canal9")))
            { //No se debe agregar un canal de resolucion BAJA a un paquete de resolcuion ALTA
                errores += "ERROR AL INGRESAR UN CANAL\n";
            }
            Paquete auxPaq2 = BuscarPaquete("paqueteSD1");
            auxPaq2.IngresarCanal(BuscarCanal("canal1"));
            auxPaq2.IngresarCanal(BuscarCanal("canal2"));
            return errores;

        }

        /// <summary>
        /// dado un nombre resolucion, multilenguaje y precio crea un canal nuevo y lo agrega a la lista de canales
        /// </summary>
        /// <param name="nombre"></param>
        /// <param name="resolucion"></param>
        /// <param name="multilenguaje"></param>
        /// <param name="precio"></param>
        /// <returns></returns>
        public Canal AgregarCanal(string nombre, Resolucion resolucion, bool multilenguaje, decimal precio)
        {
            Canal unC = null;
            if (Canal.ValidarNombre(nombre) && Canal.ValidarPrecio(precio) && BuscarCanal(nombre) == null)
            {
                unC = new Canal(nombre, resolucion, multilenguaje, precio);
                Canales.Add(unC);
            }
```

```
        return unC;
    }


    /// <summary>
    /// agregar una nueva instancia de paquete hd al sistema
    /// </summary>
    /// <param name="nombre"></param>
    /// <param name="promocion"></param>
    /// <param name="precioBase"></param>
    /// <param name="grabacionNube"></param>
    /// <param name="canales"></param>
    /// <returns></returns>
    public PaqueteHD AgregarPaqueteHD(string nombre, bool promocion, decimal
precioBase, bool grabacionNube, List<Canal> canales)
    {
        PaqueteHD unP = null;
        if (BuscarPaquete(nombre) == null)
        {
            unP = new PaqueteHD(nombre, promocion, precioBase, canales, grabacionNube);
            Paquetes.Add(unP);
        }

        return unP;
    }

    /// <summary>
    /// agregar una nueva instancia de paquete sd al sistema
    /// </summary>
    /// <param name="nombre"></param>
```

```
        /// <param name="promocion"></param>
        /// <param name="precioBase"></param>
        /// <param name="mejoraImagen"></param>
        /// <param name="canales"></param>
        /// <returns></returns>
        public PaqueteSD AgregarPaqueteSD(string nombre, bool promocion, decimal precioBase,
bool mejoraImagen, List<Canal> canales)
        {
            PaqueteSD unP = null;
            if (BuscarPaquete(nombre) == null)
            {
                unP = new PaqueteSD(nombre, promocion, precioBase, canales, mejoraImagen);
                Paquetes.Add(unP);
            }

            return unP;
        }

        /// <summary>
        /// retorna toda la lista de paquetes del sistema
        /// </summary>
        /// <returns></returns>
        public List<Paquete> ListarPaquetes()
        {
            return Paquetes;
        }

        /// <summary>
        /// retorna toda la lista de canales del sistema
        /// </summary>
        /// <returns></returns>
```

```
public List<Canal> ListarCanales()
{
   return Canales;
}

/// <summary>
/// lista los paquetes que tienen mayor cantidad de canales asociados
/// </summary>
/// <returns></returns>
public List<Paquete> ListarPaquetesConMasCanales()
{
   int mayor = int.MinValue;
   List<Paquete> aux = new List<Paquete>();

   foreach (Paquete item in Paquetes)
   {
      int cant = item.CantCanales();
      if (cant > mayor)
      {
         mayor = cant;

         aux.Clear();
         aux.Add(item);

      }
      else if (cant == mayor)
      {
         aux.Add(item);
      }
   }
```

```
    return aux;
}

/// <summary>
/// a partir de un nobmre retorna el canal asociado
/// </summary>
/// <param name="nombre"></param>
/// <returns></returns>
public Canal BuscarCanal(string nombre)
{
    Canal unC = null;
    int i = 0;
    while (unC == null && i < Canales.Count)
    {
        if (Canales[i].Nombre == nombre)
        {
            unC = Canales[i];
        }
        i++;
    }
    return unC;
}


/// <summary>
/// a partir de un nombre retorna el paquete asociado
/// </summary>
/// <param name="nombreBuscar"></param>
/// <returns></returns>
public Paquete BuscarPaquete(string nombreBuscar)
{
```

```
    Paquete unP = null;
    int i = 0;
    while (unP == null && i < Paquetes.Count)
    {
        if (Paquetes[i].Nombre == nombreBuscar)
        {
            unP = Paquetes[i];
        }
        i++;
    }
    return unP;
}

/// <summary>
/// lista los paquetes con precio mayor a "precio"
/// </summary>
/// <param name="precio"></param>
/// <returns></returns>
public List<Paquete> PaquetesMayorPrecio(decimal precioComparar)
{
    decimal max = precioComparar;
    List<Paquete> paquetesAux = new List<Paquete>();

    foreach (Paquete paqueteAux in Paquetes)
    {
        decimal precioActual = paqueteAux.PrecioBase;

        if (precioActual > max)
        {
            paquetesAux.Add(paqueteAux);
        }
```

```
            }
            return paquetesAux;
        }

        #endregion

    }
}
```

Program.cs

```
using Dominio;
using System;
using System.Collections.Generic;
using System.Linq;

namespace ObligatorioProg2
{
    class Program
    {
        private static Empresa empresa = new Empresa();
        static void Main(string[] args)
        {
            MostrarMenu();
        }

        private static void MostrarPaquetes()
        {
            Console.WriteLine("Paquetes: ");

            foreach (var v in empresa.ListarPaquetes())
            {
                Console.WriteLine(v.ToString());
            }
        }

        private static void MostrarCanales()
        {
            Console.WriteLine("Canales: ");
            foreach (var v in empresa.ListarCanales())
            {
                Console.WriteLine(v.ToString());
            }
        }

        /// <summary>
        /// Programa que se muestra en consola
        /// </summary>
        public static void MostrarMenu()
        {
            Console.WriteLine("**************************\nObligatorio -
Programación 2\n**************************");
            int salir = -1;
            do
            {
                Console.WriteLine("");
                salir = CrearMenu();

                switch (salir)
                {
                    case 1:
                        IngresarCanalTV();
                        break;
                    case 2:
                        MostrarPaquetesCanales();
                        break;
                    case 3:
                        ModificarCostoFijo();
                        break;
                    case 4:
                        MostrarPaquetesConMasCanales();
```

```
                    break;
                case 5:
                    MostrarPaquetesPrecioMayorA();
                    break;
                case 6:
                    MostrarCanales();
                    break;
                case 7:
                    MostrarPaquetes();
                    break;
                case 8:
                    salir = 0;
                    break;

            }

        } while (salir != 0);
    }

    /// <summary>
    /// Crea el menu de opciones
    /// </summary>
    /// <returns></returns>
    public static int CrearMenu()
    {
        int opcion = -1;

        Console.WriteLine("Menu:");
        Console.WriteLine("1. Ingresar un canal de TV.");
        Console.WriteLine("2. Visualizar todos los paquetes de canales.");
        Console.WriteLine("3. Modificar el costo fijo de grabación en la
nube.");
        Console.WriteLine("4. Mostrar los paquetes con la mayor cantidad de
canales.");
        Console.WriteLine("5. Listar los paquetes cuyo precio supere un valor
dado.");
        Console.WriteLine("6. Mostrar Todos los Canales");
        Console.WriteLine("7. Mostrar Todos los Paquetes");
        Console.WriteLine("8. Salir");
        Console.WriteLine("Para salir ingrese '0'.");

        opcion = int.Parse(Console.ReadLine());

        return opcion;
    }

    /// <summary>
    /// Guarda resolucion de canal
    /// </summary>
    public static void IngresarCanalTV()
    {
        string nombre = "";
        bool validar = false;
        do
        {
            //nombre, resolucion ATLA o BAJA, multilenguaje TRUE, decimal
PRECIO
            Console.WriteLine("Ingrese el nombre del canal: ");
            nombre = Console.ReadLine();

            validar = Canal.ValidarNombre(nombre);
```

```
                if (!validar)
                {
                    Console.WriteLine("El nombre del canal debe ser mayor a tres
caracteres.");
                }

            } while (!validar);

            string resolucionString = "";
            do
            {
                Console.WriteLine("Ingrese la resolucion del canal, 1080 'alta' o
576 'baja': ");
                resolucionString = Console.ReadLine();
                resolucionString = resolucionString.ToLower();
            } while (resolucionString != "alta" && resolucionString != "baja");

            Resolucion resolucionFinal =
(Resolucion)Enum.Parse(typeof(Resolucion), resolucionString.ToUpper());

            //si la respuesta es ALTA se asigna este valor a la resolucion, de no
ser asi se asigna BAJA
            //Resolucion resolucionFinal = resolucionString == "alta" ?
Resolucion.ALTA : Resolucion.BAJA;

            string multiString;
            do
            {
                Console.WriteLine("Ingrese si tiene multilenguage, 'si' o 'no':
");
                multiString = Console.ReadLine();
                multiString = multiString.ToLower();
            } while (multiString != "si" && multiString != "no");

            //si la respuesta es SI se asigna TRUE a multilenguaje, de no ser asi
se asigna FALSE
            bool multiFinal = multiString == "si" ? true : false;
            decimal precioDecimal = -1;
            do
            {
                precioDecimal = GuardarValorDecimal("Ingrese el precio del mismo:
");

                if (!Canal.ValidarPrecio(precioDecimal))
                {
                    Console.WriteLine("Ingeese un precio de Canal mayo a 0.");
                }
            } while (!Canal.ValidarPrecio(precioDecimal));

            if (Canal.ValidarPrecio(precioDecimal) &&
Canal.ValidarNombre(nombre)) { }
            {
                Canal unC = empresa.AgregarCanal(nombre, resolucionFinal,
multiFinal, precioDecimal);
                Console.WriteLine("Su canal ha sido creado.");
                Console.WriteLine(unC);
            }

        }

        /// <summary>
        /// Muestra paquetes de canales
```

```
        /// </summary>
        public static void MostrarPaquetesCanales()
        {
            Console.WriteLine("Lista de todos los paquetes de canales.");

            List<Paquete> paquetesLista = empresa.ListarPaquetes();
            foreach (Paquete item in paquetesLista)
            {
                Console.WriteLine(item);
            }

        }

        /// <summary>
        /// Modifica el costo fijo de grabacion en la nube
        /// </summary>
        public static void ModificarCostoFijo()
        {
            Console.WriteLine($"Costo fijo de grabacion en la nube actual:
{PaqueteHD.CostoFijo}");

            decimal precioACambiar = GuardarValorDecimal("Ingresa un nuevo valor
para el costo fijo de grabación en la nube: ");

            if (!PaqueteHD.ValidarPrecioNube(precioACambiar))
            {
                Console.WriteLine("Precio de grabacion en la nube no valido");
            }
            else
            {
                PaqueteHD.CostoFijo = precioACambiar;

                Console.WriteLine("Precio cambiado exitosamente.");

            }
        }

        /// <summary>
        /// Muestra paquetes con mas canales
        /// </summary>
        public static void MostrarPaquetesConMasCanales()
        {
            List<Paquete> listaaux = empresa.ListarPaquetesConMasCanales();

            foreach (Paquete item in listaaux)
            {
                Console.WriteLine($"Cant Canales: {item.Canales.Count}");
                Console.WriteLine(item);
            }
        }

        /// <summary>
        /// Muestra los paquetes con mayor precio que el comparado ingresado por
el usuario
        /// </summary>
        public static void MostrarPaquetesPrecioMayorA()
        {
            decimal precioComparar = GuardarValorDecimal("Ingrese un precio a
comparar: ");
            List<Paquete> listaaux = empresa.PaquetesMayorPrecio(precioComparar);
            foreach (Paquete item in listaaux)
            {
```

```
                Console.WriteLine(item);
            }
        }

        /// <summary>
        /// Metodo auxiliar que guarda el valor decimal y muestra el mensaje
pasado por parametro
        /// </summary>
        /// <param name="mensaje"></param>
        /// <returns></returns>
        public static decimal GuardarValorDecimal(string mensaje)
        {

            decimal nuevoprecio = -1;
            string precioString = "";
            bool exito = false;
            do
            {
                Console.WriteLine(mensaje);
                precioString = Console.ReadLine();
                exito = decimal.TryParse(precioString, out nuevoprecio);

            } while (!exito);

            return nuevoprecio;
        }
    }
}
```