Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO 1 PROGRAMACION 2

DOCUMENTO DE ANÁLISIS

Grupo M2A

Gabriel Abelenda – 233401

Maximiliano Raimondo – 249003

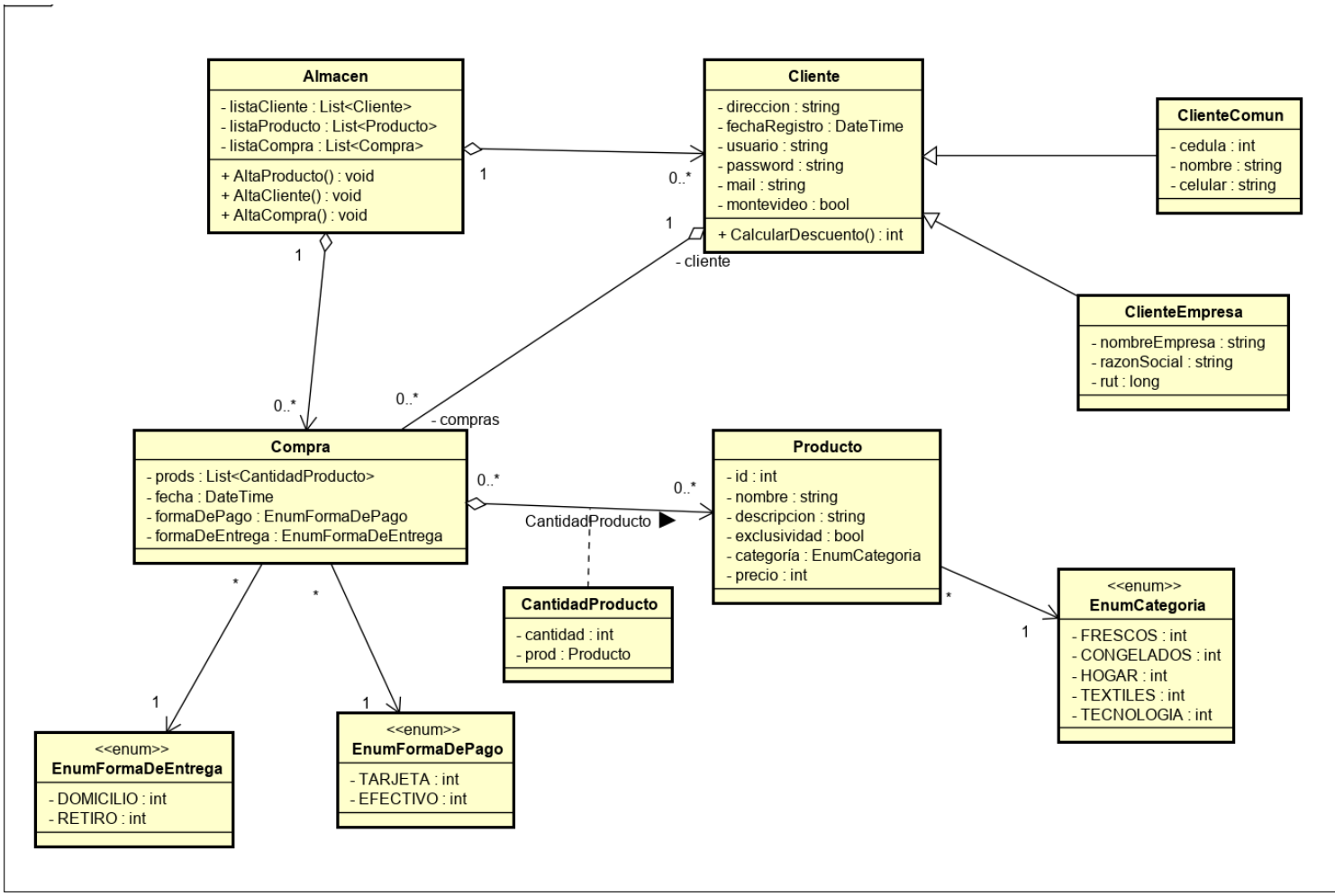
M2A

Docente: Francisco Bouza

Analista en Tecnologías de la Información/Analista Programador

(17-10-2019)

1. Diagrama de clases



2. Precarga de datos y código fuente.

```
**********************************
Filename: Almacen.cs
**********************************
using System;
using System.Collections.Generic;
using System.Text;

namespace Dominio
{
   public class Almacen
   {
      private List<Cliente> clientes;
      private List<Producto> productos;
      private List<Compra> compras;

      #region Singleton
      private static Almacen instancia;

      public static Almacen Instancia
      {
         get
         {
            if (instancia == null)
            {
               instancia = new Almacen();
            }
            return instancia;
         }
```

```
}
#endregion



#region Properties
public List<Cliente> Clientes
{
    get { return clientes; }
    set { clientes = value; }
}



public List<Producto> Productos
{
    get { return productos; }
    set { productos = value; }
}



public List<Compra> Compras
{
    get { return compras; }
    set { compras = value; }
}
#endregion



private Almacen()
{
    this.clientes = new List<Cliente>();
```

```
            this.productos = new List<Producto>();
            this.compras = new List<Compra>();
        }

        public void AltaProducto(Producto p)
        {
            productos.Add(p);
        }

        public void AltaCliente(Cliente c)
        {
            clientes.Add(c);
        }
        public void AltaCompra(Compra co)
        {
            compras.Add(co);
        }
    }
}
```

****

Filename: CantidadProducto.cs

****

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

namespace Dominio
{
   public class CantidadProducto
    {
```

```
        private int cantidad;
        private Producto prod;


        #region Properties
        public int Cantidad
        {
            get { return cantidad; }
            set { cantidad = value; }
        }


        public Producto Prod
        {
            get { return prod; }
            set { prod = value; }
        }
        #endregion

        public CantidadProducto(int cantidad, Producto prod)
        {
            this.cantidad = cantidad;
            this.prod = prod;
        }
    }
}
```

****

Filename: Cliente.cs

****

```
using System;
```

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

namespace Dominio
{
    public class Cliente
   {
       private string direccion;
       private DateTime fechaRegistro;
       private string usuario;
       private string password;
       private string mail;
       private bool montevideo;
       private List<Compra> compras;


       #region Properties
       public List<Compra> Compras
       {
          get { return compras;}
          set { compras = value;}
       }

       public string Direccion
       {
          get { return direccion; }
          set { direccion = value; }
       }


       public DateTime FechaRegistro
```

```
{
    get { return fechaRegistro; }
    set { fechaRegistro = value; }
}

public string Usuario
{
    get { return usuario; }
    set { usuario = value; }
}

public string Password
{
    get { return password; }
    set { password = value; }
}

public string Mail
{
    get { return mail; }
    set { mail = value; }
}

public bool Montevideo
{
    get { return montevideo; }
    set { montevideo = value; }
```

```
        }

        #endregion

        public Cliente(string direccion, DateTime fechaRegistro, string usuario, string password, string mail, bool
montevideo)
        {
            this.direccion = direccion;
            this.fechaRegistro = fechaRegistro;
            this.usuario = usuario;
            this.password = password;
            this.mail = mail;
            this.montevideo = montevideo;
            this.compras = new List<Compra>();

        }




    }
}
```

****

Filename: ClienteComun.cs

****

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

namespace Dominio
```

```
{
  public class ClienteComun : Cliente

   {
      private int cedula;
      private string nombre;
      private string celular;
      private string tipoCliente;

      #region Properties
      public int Cedula
      {
         get { return cedula; }
         set { cedula = value; }
      }

      public string Nombre
      {
         get { return nombre; }
         set { nombre = value; }
      }


      public string Celular
      {
         get { return celular; }
         set { celular = value; }
      }

      public string TipoCliente
      {
```

```
        get { return tipoCliente; }
        set { tipoCliente = value; }
    }

    #endregion

    public ClienteComun(string direccion, DateTime fechaRegistro, string usuario, string password, string mail, bool montevideo, int cedula, string nombre, string celular)
        :base(direccion, fechaRegistro, usuario, password, mail, montevideo)
    {

        this.cedula = cedula;
        this.nombre = nombre;
        this.celular = celular;
        this.TipoCliente = "Común";
    }

    public override string ToString()
    {
        return string.Format("Nombre: {0}, Email: {1} (Tipo de cliente: {2})", nombre,Mail,tipoCliente);
    }

  }
}
```

****

Filename: ClienteEmpresa.cs

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

namespace Dominio
{
    public class ClienteEmpresa : Cliente
    {
        private string nombreEmpresa;
        private string razonSocial;
        private long rut;
        private string tipoCliente;

        #region Properties
        public string NombreEmpresa
        {
            get { return nombreEmpresa; }
            set { nombreEmpresa = value; }

        }

        public string RazonSocial
        {
            get { return razonSocial; }
            set { razonSocial = value; }
        }


        public long Rut
        {
```

```
        get { return rut; }
        set { rut = value; }
    }

    public string TipoCliente
    {
        get { return tipoCliente; }
        set { tipoCliente = value; }
    }

    #endregion

    public ClienteEmpresa(string direccion, DateTime fechaRegistro, string usuario, string password, string mail,
bool montevideo, string nombreEmpresa, string razonSocial, long rut)
        :base(direccion, fechaRegistro, usuario, password, mail, montevideo)
    {
        this.nombreEmpresa = nombreEmpresa;
        this.razonSocial = razonSocial;
        this.rut = rut;
        this.tipoCliente = "Empresa";
    }

    public override string ToString()
    {
        return string.Format("Nombre: {0}, Email: {1} (Tipo de cliente: {2}) ", nombreEmpresa, Mail, tipoCliente);
    }
```

```
    }
}
```

****

Filename: Compra.cs

****

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

namespace Dominio
{
  public class Compra
   {
      #region Definicion Enums
      public enum EnumFormaDePago
      {
         TARJETA = 1,
         EFECTIVO
      }

      public enum EnumFormaDeEntrega
      {
         DOMICILIO = 1,
         RETIRO
      }

      #endregion

      private List<CantidadProducto> prods;
      private DateTime fecha;
```

```
private EnumFormaDePago formaDePago;
private EnumFormaDeEntrega formaDeEntrega;
private Cliente cliente;

#region Properties

public Cliente Cliente
{
   get { return cliente; }
   set { cliente = value; }
}

public List<CantidadProducto> Prods
{
   get { return prods; }
   set { prods = value; }
}

public DateTime Fecha
{
   get { return fecha; }
   set { fecha = value; }
}


public EnumFormaDePago FormaDePago
{
   get { return formaDePago; }
}
```

```
        public EnumFormaDeEntrega FormaDeEntrega
        {
            get { return formaDeEntrega; }
            set { formaDeEntrega = value; }
        }
        #endregion

        public Compra( DateTime fecha, EnumFormaDePago formaDePago, EnumFormaDeEntrega formaDeEntrega,
Cliente cliente)
        {

            this.prods = new List<CantidadProducto>();
            this.fecha = fecha;
            this.formaDePago = formaDePago;
            this.formaDeEntrega = formaDeEntrega;
            this.cliente = cliente;
        }

        public void AltaCantidadProducto(CantidadProducto cp)
        {
            prods.Add(cp);
        }

        public override string ToString()
        {
            return string.Format("Usuario de cliente: {0}, Direccion de cliente: {1}, Fecha de compra: {2}, Cantidad de
productos: {3} ", cliente.Usuario, cliente.Direccion, fecha.ToShortDateString(), prods.Count);
        }

    }
}
```

```
************************************
Filename: Producto.cs
************************************
using System;
using System.Collections.Generic;
using System.Text;

namespace Dominio
{
    public class Producto
    {
        #region Definicion Enums

        public enum EnumCategoria
        {
            FRESCOS = 1,
            CONGELADOS,
            HOGAR,
            TEXTILES,
            TECNOLOGIA
        }

        #endregion

        private static int ultId = 1;
        private int id;
        private string nombre;
        private string descripcion;
        private bool exclusividad;
        private EnumCategoria categoria;
```

```
private int precio;

public Producto(string nombre, string descripcion, bool exclusividad, EnumCategoria categoria, int precio)
{
    this.id = ultId++;
    this.nombre = nombre;
    this.descripcion = descripcion;
    this.exclusividad = exclusividad;
    this.categoria = categoria;
    this.precio = precio;
}

#region Properties
public int ID

{
    get { return id; }
    set { id = value; }
}


public string Nombre
{
    get { return nombre; }
    set { nombre = value; }
}


public string Descripcion
{
    get { return descripcion; }
```

```
    set { descripcion = value; }
}

public bool Exclusividad
{
    get { return exclusividad; }
    set { exclusividad = value; }
}

public EnumCategoria Categoria
{
    get { return categoria; }
}

public int Precio
{
    get { return precio; }
    set { precio = value; }
}
#endregion

public override string ToString()
{
    return string.Format("{0}: {1} {2} (${3})" ,id , nombre, categoria, precio);
}
```

```
    }
}
```

****

Filename: Program.cs

****

```
using Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace ObligatorioP2
{
    class Program
    {
        static void Main(string[] args)

        {
            try
            {
                string rootPath =
System.IO.Path.GetDirectoryName(System.IO.Path.GetDirectoryName(System.IO.Path.GetDirectoryName(System.I
O.Directory.GetCurrentDirectory())));
                var header = "**********" + Environment.NewLine;

                var files = System.IO.Directory.GetFiles(rootPath, "*.cs", System.IO.SearchOption.AllDirectories);
```

```
        var result = files.Where(p => !p.Contains("Temporary") && !p.Contains("AssemblyInfo.cs")).Select(path
=> new { Name = System.IO.Path.GetFileName(path), Contents = System.IO.File.ReadAllText(path) })

                        .Select(info =>

                          header

                        + "Filename: " + info.Name + Environment.NewLine

                        + header

                        + info.Contents);



        var singleStr = string.Join(Environment.NewLine, result);

System.IO.File.WriteAllText(System.IO.Path.GetDirectoryName(System.IO.Path.GetDirectoryName(System.IO.Path
.GetDirectoryName(System.IO.Directory.GetCurrentDirectory()))) + @"\output.txt", singleStr, Encoding.UTF8);

        }

        catch (Exception algunError)

        {

           Console.WriteLine(algunError.Message);

        }

        //CARGA DE PRODUCTOS;

        Producto p1 = new Producto("Pescado", "Atún en lata", false, Producto.EnumCategoria.FRESCOS, 50);

        Producto p2 = new Producto("Cafetera", "Recargable capuccino", false,
Producto.EnumCategoria.TECNOLOGIA, 2000);

        Producto p3 = new Producto("Mix verduras", "Verduras congeladas", false,
Producto.EnumCategoria.CONGELADOS, 150);

        Producto p4 = new Producto("Alfombra", "Alfombra con decoración", true,
Producto.EnumCategoria.TEXTILES, 800);

        Producto p5 = new Producto("Juego de living", "4 sillas y una mesa ", false,
Producto.EnumCategoria.HOGAR, 5000);

        Producto p6 = new Producto("Philips Smart Tv 65'", "Tv 4k marca Philips", true,
Producto.EnumCategoria.TECNOLOGIA, 25000);



        //CARGA DE CLIENTES;

        Cliente c1 = new ClienteComun("Union 123", Fecha("12/04/2019"), "jorgito123", "1234",
"jorgito@gmail.com", false, 1234567, "Jorge", "099566411");
```

```
Cliente c2 = new ClienteComun("Joaquin Suarez 354", Fecha("19/07/2018"), "pedro543", "1234", "Manya@gmail.com", true, 7654321, "Pedro", "098766355");

Cliente c3 = new ClienteComun("18 de Julio 678", Fecha("05/09/2019"), "martucho123", "1234", "NTVG@gmail.com", false, 8796541, "Martín", "091765411");

Cliente c4 = new ClienteEmpresa("Colonia 345", Fecha("16/09/2017"), "axion12", "1234", "axion@petrolera.com", false, "AXION", "Axion S.A.", 8765431222);

Cliente c5 = new ClienteEmpresa("San Jose 432", Fecha("22/03/2018"), "BarDeMoe12", "1234", "moe@bar.com", true, "Moe's", "Duff S.A.", 6574732151);

Cliente c6 = new ClienteEmpresa("Boulevard 897", Fecha("07/02/2019"), "TecnoMundo77", "1234", "TecMundo@tecnomundo.com", true, "Tecnomundo", "Mier S.R.L", 8765431222);


//CARGA DE CANTIDADES DE PRODUCTOS;

CantidadProducto cp1 = new CantidadProducto(2, p3);

CantidadProducto cp2 = new CantidadProducto(1, p1);

CantidadProducto cp3 = new CantidadProducto(5, p2);

CantidadProducto cp4 = new CantidadProducto(2, p4);

CantidadProducto cp5 = new CantidadProducto(1, p5);

CantidadProducto cp6 = new CantidadProducto(2, p6);

CantidadProducto cp7 = new CantidadProducto(4, p1);

CantidadProducto cp8 = new CantidadProducto(1, p6);


//CARGA Y ALTA DE COMPRAS;

Compra compra1 = new Compra(Fecha("20/05/2019"), Compra.EnumFormaDePago.TARJETA, Compra.EnumFormaDeEntrega.DOMICILIO, c4);

compra1.AltaCantidadProducto(cp2);

compra1.AltaCantidadProducto(cp1);

compra1.AltaCantidadProducto(cp4);

c4.Compras.Add(compra1);

Almacen.Instancia.AltaCompra(compra1);


Compra compra2 = new Compra(Fecha("31/07/2019"), Compra.EnumFormaDePago.EFECTIVO, Compra.EnumFormaDeEntrega.DOMICILIO, c1);

compra2.AltaCantidadProducto(cp3);

compra2.AltaCantidadProducto(cp6);
```

```
        compra2.AltaCantidadProducto(cp4);
        c1.Compras.Add(compra2);
        Almacen.Instancia.AltaCompra(compra2);

        Compra compra3 = new Compra(Fecha("25/05/2018"), Compra.EnumFormaDePago.EFECTIVO,
Compra.EnumFormaDeEntrega.RETIRO, c2);
        compra3.AltaCantidadProducto(cp4);
        compra3.AltaCantidadProducto(cp7);
        c2.Compras.Add(compra3);
        Almacen.Instancia.AltaCompra(compra3);

        Compra compra4 = new Compra(Fecha("20/05/2017"), Compra.EnumFormaDePago.TARJETA,
Compra.EnumFormaDeEntrega.RETIRO, c3);
        compra4.AltaCantidadProducto(cp5);
        compra4.AltaCantidadProducto(cp8);
        compra4.AltaCantidadProducto(cp3);
        c3.Compras.Add(compra4);
        Almacen.Instancia.AltaCompra(compra4);

        //ALTAS DE PRODUCTOS
        Almacen.Instancia.AltaProducto(p1);
        Almacen.Instancia.AltaProducto(p2);
        Almacen.Instancia.AltaProducto(p3);
        Almacen.Instancia.AltaProducto(p4);
        Almacen.Instancia.AltaProducto(p5);
        Almacen.Instancia.AltaProducto(p6);
        Almacen.Instancia.AltaCliente(c1);
        Almacen.Instancia.AltaCliente(c2);
        Almacen.Instancia.AltaCliente(c3);
        Almacen.Instancia.AltaCliente(c4);
        Almacen.Instancia.AltaCliente(c5);
        Almacen.Instancia.AltaCliente(c6);
```

```
    Menu();



}



private static void Menu()
{

    Console.ForegroundColor = ConsoleColor.Yellow;
    bool sigo = true;
    do
    {
        Console.Clear();
        Console.WriteLine("1- Listar Catálogo de Productos");
        Console.WriteLine("2- Listar Clientes");
        Console.WriteLine("3- Listar Compras");
        Console.WriteLine("4- Agregar Producto");
        Console.WriteLine("5- Salir");
        int opcion = PedirNumero("Ingrese una opción:", "La opción debe estar entre 1 y 5", 1, 5);
        switch (opcion)
        {
            case 1:
                ListarProducto();
                break;
            case 2:
                ListarClientes();
                break;
```

```
            case 3:
                ListarCompras();
                break;
            case 4:
                AgregarProducto();
                break;

            case 5:
                sigo = false;
                Console.WriteLine("CHAUCHA");
                Console.ReadKey();
                break;
            default:
                break;
        }
    } while (sigo);
}
private static void ListarProducto()
{
    bool sigo = true;
    do
    {
        Console.WriteLine("1- Frescos");
        Console.WriteLine("2- Congelados");
        Console.WriteLine("3- Hogar");
        Console.WriteLine("4- Textiles");
        Console.WriteLine("5- Tecnología");
        Console.WriteLine("6- Todos");
        Console.WriteLine("7- Salir");
        int opcion = PedirNumero("Ingrese el número de la categoría", "Ingrese el número correctamente", 1, 7);
        if (opcion == 7)
```

```
{
    sigo = false;
}
else if (opcion == 6)
{
    if (Almacen.Instancia.Productos.Count == 0)
    {
        Console.WriteLine("No hay productos.");
    }
    else
    {
        foreach (Producto prod in Almacen.Instancia.Productos)
        {
            Console.WriteLine(prod);


        }
        Console.ReadKey();
    }
}
else
{


    foreach (Producto prod in Almacen.Instancia.Productos)
    {
        if (prod.Categoria == (Producto.EnumCategoria)opcion)
        {
            Console.WriteLine(prod);
        }
    }
    Console.ReadKey();
}
```

```
        } while (sigo);


    }
    private static void ListarClientes()
    {
        //PIDO FECHA Y BUSCO
        DateTime fechaIngresada = PedirFecha("Ingrese una fecha en el siguiente formato: dd/MM/yyyy", "El formato no es correcto", "dd/MM/yyyy");
        bool hayClientes = false;
        foreach (Cliente cli in Almacen.Instancia.Clientes)
        {
            if (cli.FechaRegistro < fechaIngresada)
            {
                Console.WriteLine(cli);
                hayClientes = true;     //VERIFICO EXISTENCIA
            }
        }
        if (!hayClientes)
        {
            Console.WriteLine("No se encontraron cliente registrados antes de la fecha ingresada");
        }
        Console.ReadKey();


    }




    private static void ListarCompras()
    {
        //PIDO FECHAS
        DateTime fechaIngresada1 = PedirFecha("Ingrese una fecha de inicio de búsqueda de compras realizadas en el siguiente formato: dd/MM/yyyy", "El formato no es correcto", "dd/MM/yyyy");
```

```
        DateTime fechaIngresada2 = PedirFecha("Ingrese una fecha de fin de búsqueda de compras realizadas en el siguiente formato: dd/MM/yyyy", "El formato no es correcto", "dd/MM/yyyy");
        bool hayCompras = false;
        //BUSCO COMPRAS ENTRE ESAS FECHAS
        foreach (Compra com in Almacen.Instancia.Compras)
        {
            if (com.Fecha >= fechaIngresada1 && com.Fecha <= fechaIngresada2)
            {
                Console.WriteLine(com);
                hayCompras = true; //VERIFICO EXISTENCIA
            }
        }
        if (!hayCompras)
        {
            Console.WriteLine("No se encontraron compras registradas entre las fechas ingresadas");
        }
        Console.ReadKey();



    }







    //PARSEO LA FECHA
    private static DateTime Fecha(string f)
    {
        DateTime pFecha = DateTime.ParseExact(f, "dd/MM/yyyy", null);
        pFecha.ToString("dd/MM/yyyy");
        return pFecha;
    }
```

```
//IMPRIMO CATEGORIAS Y PIDO NUMERO
private static int PedirCategoria()
{
    Console.WriteLine("1- Frescos");
    Console.WriteLine("2- Congelados");
    Console.WriteLine("3- Hogar");
    Console.WriteLine("4- Textiles");
    Console.WriteLine("5- Tecnología");
    int opcion = PedirNumero("Seleccione una categoría", "Ingrese el número correctamente", 1, 5);
    return opcion;
}

private static DateTime PedirFecha(string msg, string errMsg, string formato)
{
    bool exito = false;
    DateTime fecha;
    do
    {
        Console.WriteLine(msg);
        string fechaStr = Console.ReadLine();
        exito = DateTime.TryParseExact(fechaStr, formato, null, System.Globalization.DateTimeStyles.None, out
fecha);
        if (!exito)
            Console.WriteLine(errMsg);
    } while (!exito);
    return fecha;
}

private static void AgregarProducto()
{
```

```
        //VERIFICO EXISTENCIA
        string nombre = "";
        bool sigo;
        do
        {
            nombre = PedirString("Ingrese nombre del producto.", "El nombre no puede ser vacío.");
            sigo = false;
            foreach (Producto prod in Almacen.Instancia.Productos)
                if (prod.Nombre.ToLower() == nombre.ToLower())
                {
                    Console.WriteLine("El producto ya existe");
                    Console.ReadKey();
                    sigo = true;
                }
        }
        while (sigo);

        string descripcion = PedirString("Ingrese una descripción.", "La descripción no puede ser vacía.");
        bool exclusivo = PedirBool("Ingrese 'si' si el producto es exclusivo o 'no' en caso contrario.", "Ingrese de la manera indicada anteriormente."); //PIDO BOOL PARA EXCLUSIVO
        Producto.EnumCategoria categoria = (Producto.EnumCategoria)PedirCategoria(); //CASTEO CATEGORIA
        int precio = PedirNumero("Ingrese el precio del producto", "Debe ingresar un valor numérico adecuado.", 0, 99999);

        Producto p = new Producto(nombre, descripcion, exclusivo, categoria, precio);
        Almacen.Instancia.AltaProducto(p); //AGREGO PRODUCTO AL CATALOGO
        Console.WriteLine("El producto se añadió al catálogo correctamente");
        Console.ReadKey();
    }

    private static string PedirString(string msg, string errMsg)
    {
```

```
    bool exito = false;
    string str;
    do
    {
        Console.WriteLine(msg);
        str = Console.ReadLine();
        exito = str.Length > 0;
        if (!exito)
            Console.WriteLine(errMsg);
    } while (!exito);
    return str;
}
private static bool PedirBool(string msg, string errMsg)
{
    bool cumple = false;
    string str;
    Console.WriteLine(msg);
    do
    {
        str = Console.ReadLine();
        if (str == "si")
        {
            cumple = true;
        }
        else if (str != "no")
        {
            Console.WriteLine(errMsg);
        }
    } while (!(str == "si" || str == "no"));
    return cumple;
}
```

```
    private static int PedirNumero(string msg, string errMsg, int min, int max)
    {
        bool exito = false;
        int num;
        do
        {
            Console.WriteLine(msg);
            string numMsg = Console.ReadLine();
            exito = int.TryParse(numMsg, out num) && num >= min && num <= max;
            if (!exito)
                Console.WriteLine(errMsg);
        } while (!exito);
        return num;
    }

  }
}
```