# Universidad ORT Uruguay
# Facultad de Ingeniería
## Escuela de Tecnología

# OBLIGATORIO PROGRAMACION 2

# DOCUMENTACION

**Nicolas Del Core – 212764** - **Favio Reinaldo – 158891** - **Juan Zabalia – 270539**

**Grupo: N2A**

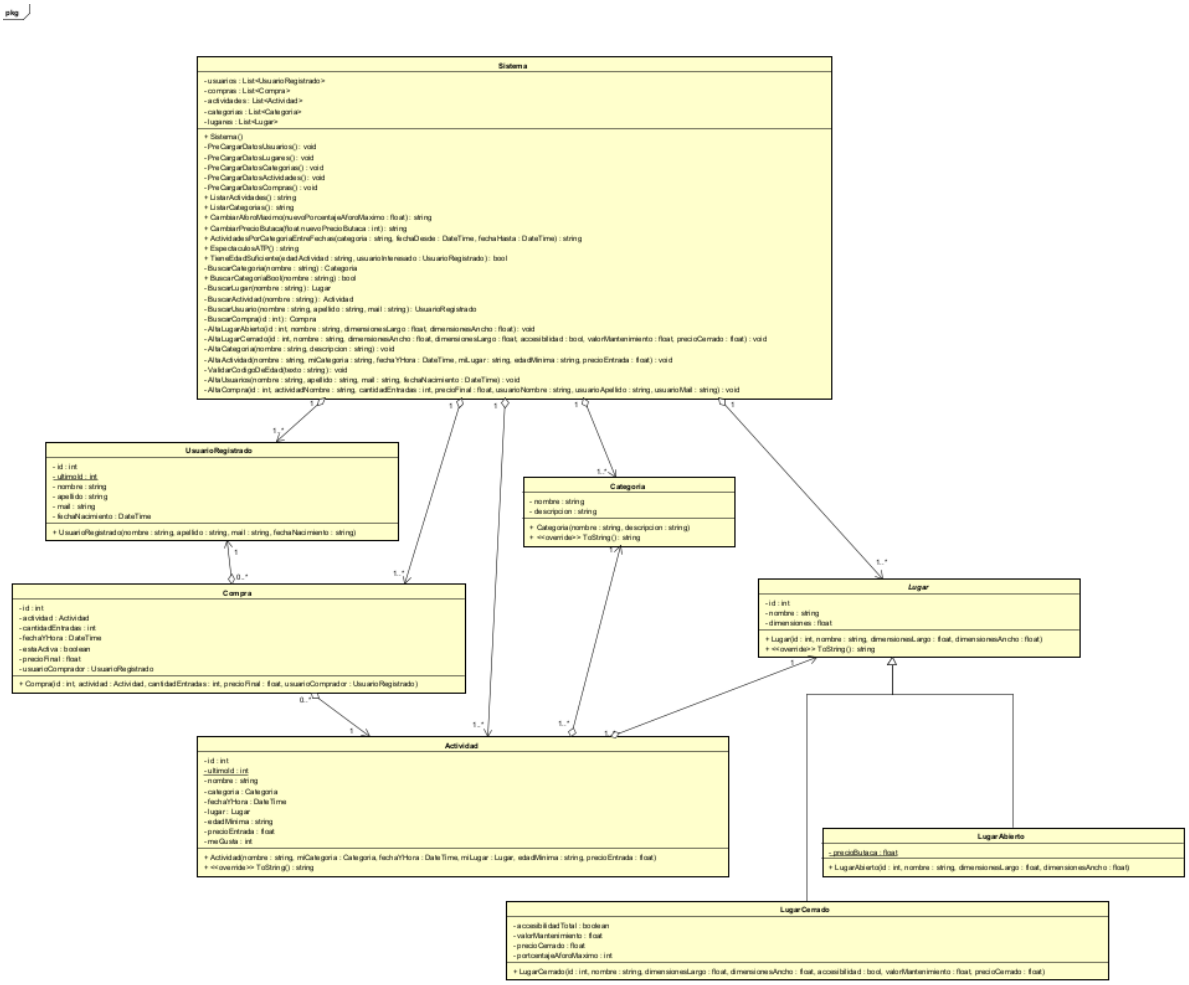
**Docente: Liliana Pino**

**Analista Programador**

**07 de Octubre de 2021**

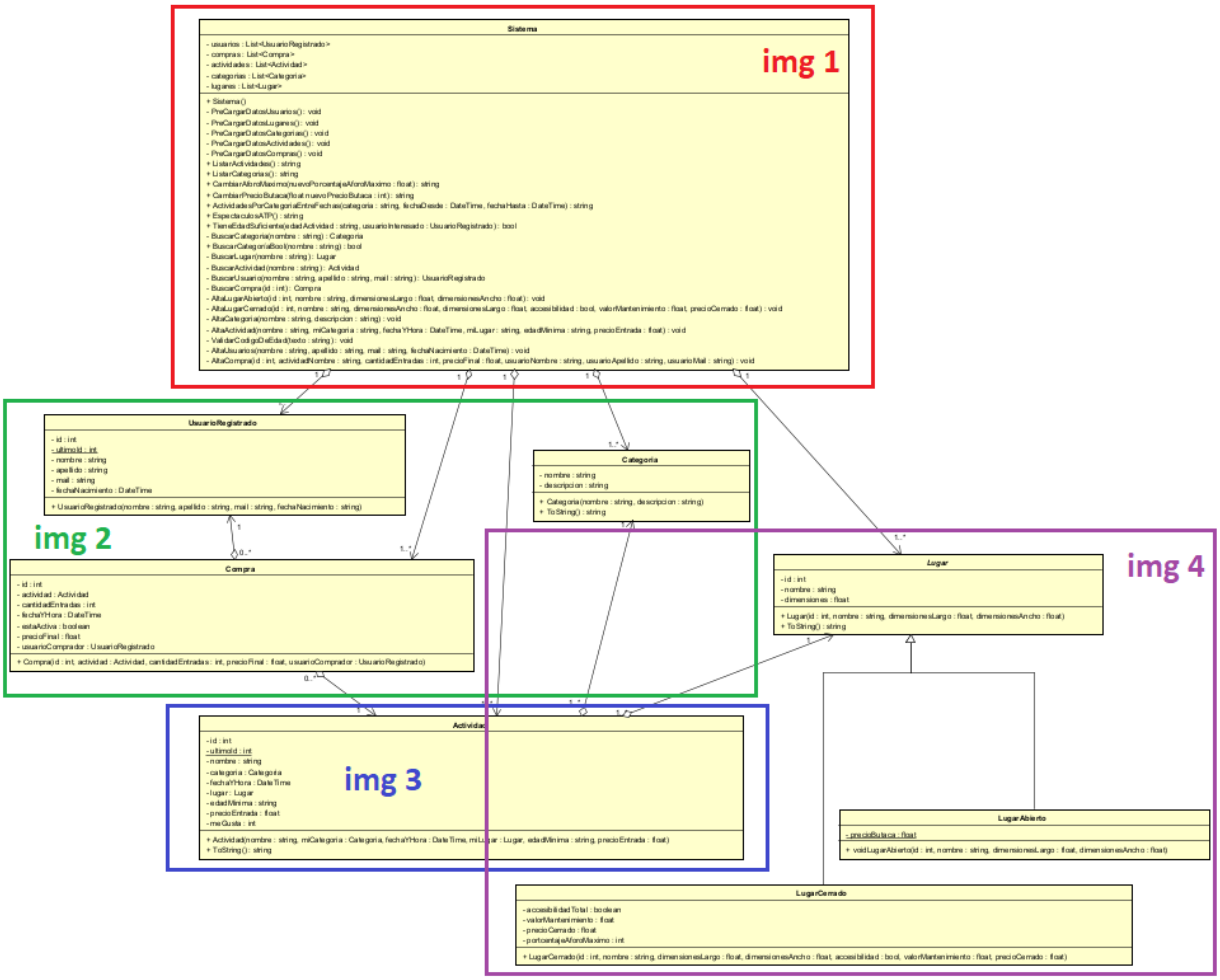
# CONTENIDOS

# 1 DIAGRAMA DE CLASES

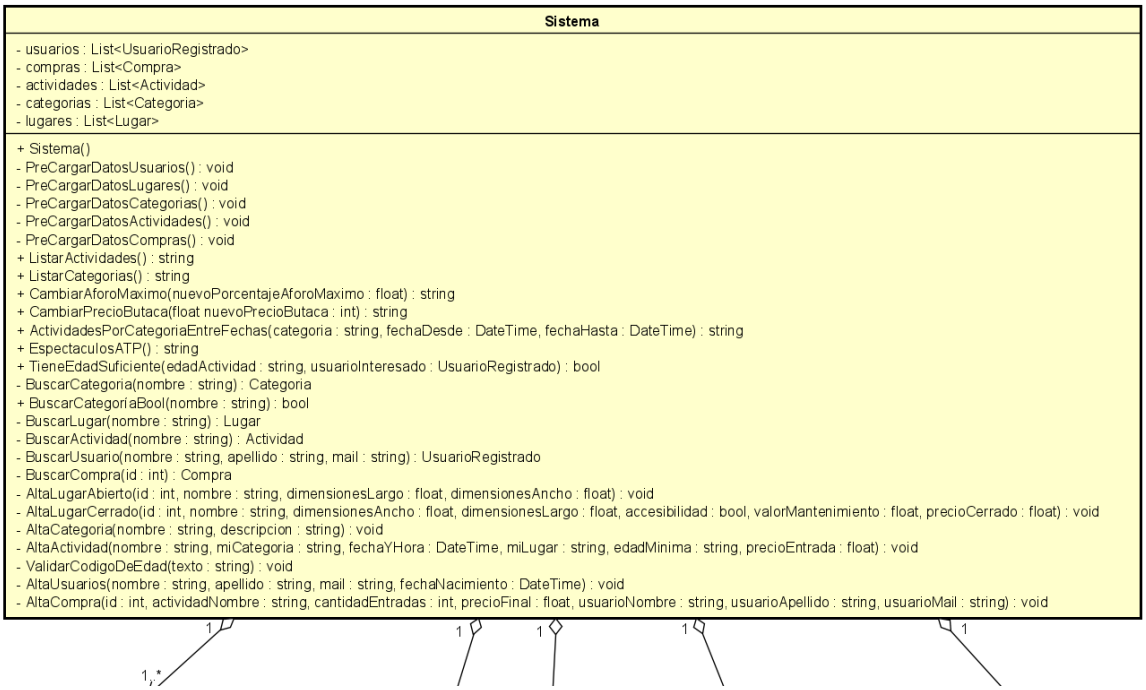



Detalle del Zoom

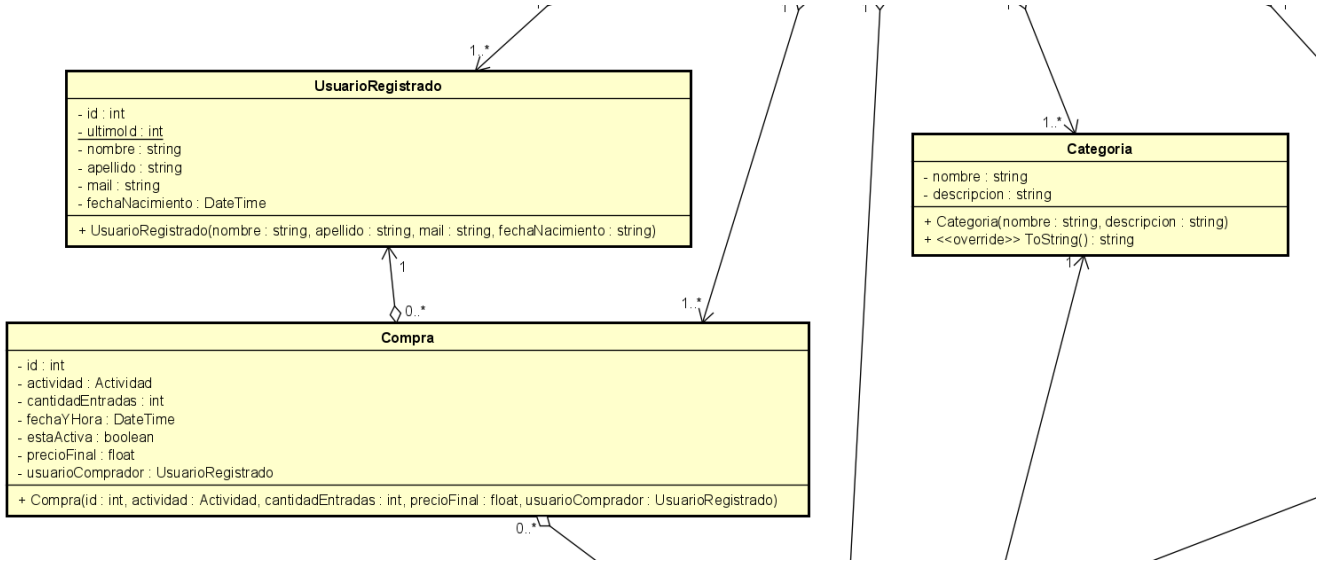
pkg



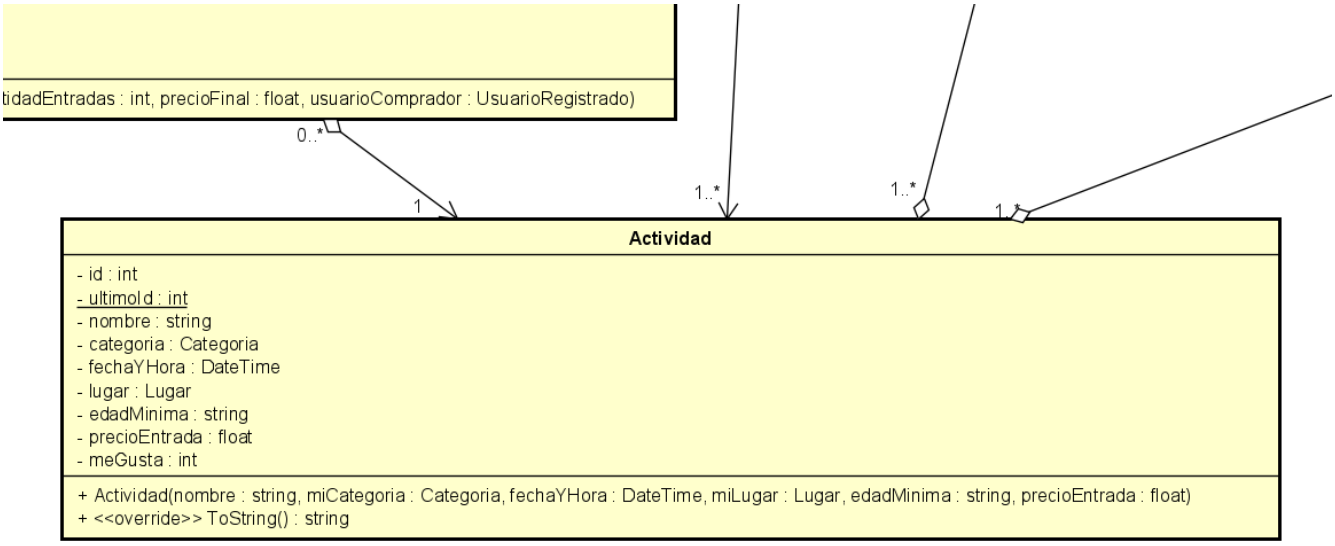

Primer recuadro (rojo) Img 1)



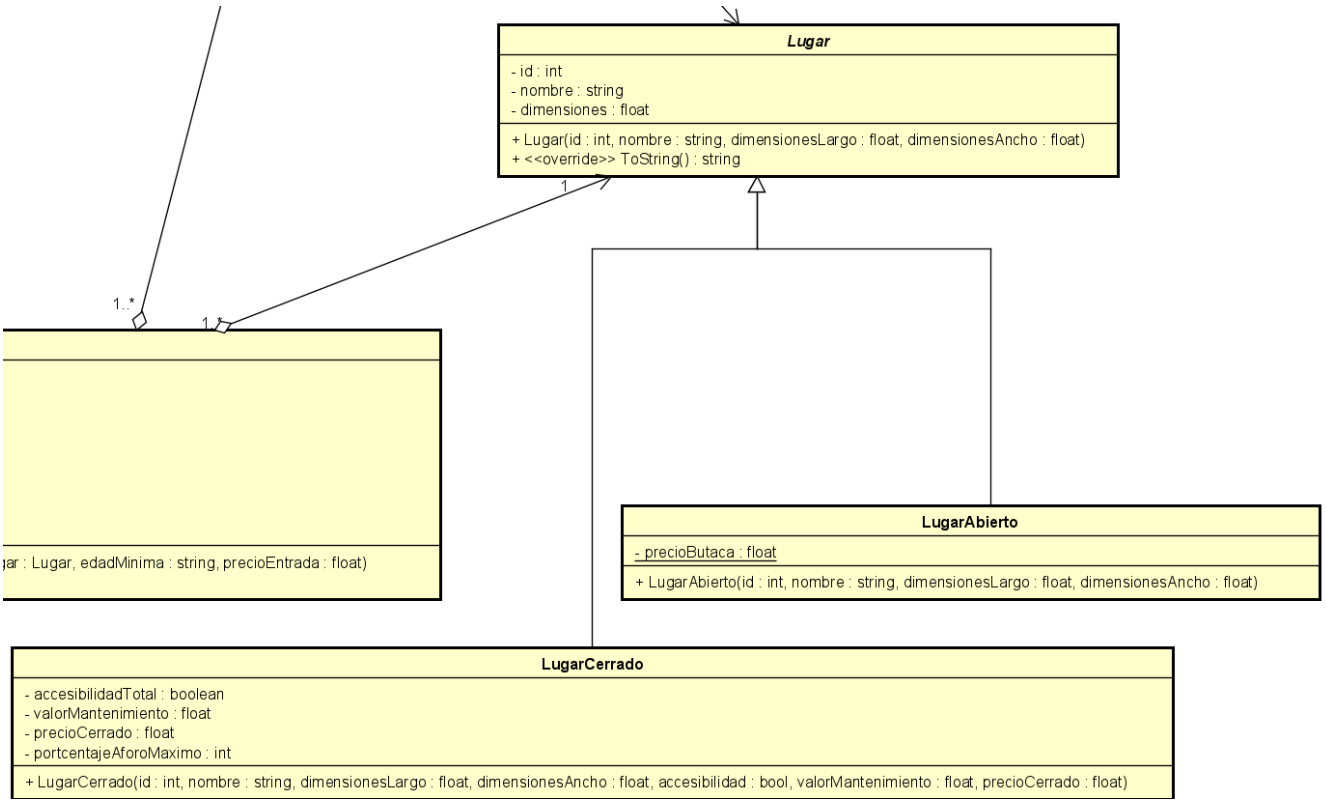

Segundo recuadro (verde) Img 2)




Tercer recuadro (azul) Img 3)




Cuarto recuadro (violeta) Img 4)



# 2 TABLA DE INFORMACION PRECARGADA

## 2.1 TABLA DE DATOS DE USUARIOS

| Nombre | Apellido | Mail | Fecha |
|---|---|---|---|
| Fernando | Fernandez | fernanfernan1@skynet.net | DateTime.Now |
| Robin | Hood | comida@paralaplebe.org | DateTime.Now |
| Es | Talin | comunista104@pueblo.org | DateTime.Now |
| Fernando | Fernandez | fefefer@skynet.net | DateTime.Now |
| Pepe | Mujica | elpepe@presidencia.uy | DateTime.Now |

## 2.2 TABLA DE DATOS DE LUGARES ABIERTOS

| ID | Nombre | DimensionAncho | DimensionLargo |
|---|---|---|---|
| 1 | Plaza Césamo | 400 | 500 |
| 2 | El Parque | 400 | 500 |
| 3 | La Plaza | 400 | 500 |
| 4 | Centenario | 400 | 500 |

## 2.3 TABLA DE DATOS DE LUGARES CERRADOS

| ID | Nombre | DimensionAncho | DimensionLargo | Accesibilidad | ValorMantenimiento | Precio |
|---|---|---|---|---|---|---|
| 5 | Intendencia | 400 | 500 | true | 700 | 999 |
| 6 | Castillo Pittamiglio | 400 | 500 | false | 700 | 999 |
| 7 | Latu | 400 | 500 | true | 700 | 999 |
| 8 | El Galpón | 400 | 500 | false | 700 | 999 |

## 2.4 TABLA DE DATOS DE CATEGORIAS

| Nombre | Descripción |
|---|---|
| Fiesta | Eventos bailables para borrachines |
| Deporte | Evento de actividad física |
| Competencia | Evento en el que se disputa un premio |
| Educación | Evento de culturización |
| Cine | Proyeccion de películas |
| Teatro | Representación de obras |
| Concierto | Evento musical en vivo |
| Feria Gastronómica | Mercado de alimentos y/o bebidas |

## 2.5 TABLA DE DATOS DE ACTIVIDADES

| Nombre | Categoria | Fecha | Lugar | EdadMinima | Precio |
|---|---|---|---|---|---|
| Fiesta de la Cerveza | Fiesta | new DateTime(2021, 10, 5, 20, 0, 0) | Plaza Césamo | C18 | 500 |
| Juntada en la Intendencia | Deporte | new DateTime(2022, 1, 24, 16, 30, 0) | Intendencia | C16 | 500 |
| Competencia de cartas yugi | Competencia | new DateTime(2021, 11, 15, 12, 30, 0) | El Parque | C13 | 500 |
| Convención del Latu | Educación | new DateTime(2021, 10, 9, 16, 30, 0) | Latu | P | 500 |
| Paseo por el Castillo | Educación | new DateTime(2021, 12, 15, 10, 30, 0) | Castillo Pittamiglio | C13 | 500 |
| Matrix Resurrections | Cine | new DateTime(2021, 10, 10, 18, 45, 0) | El Galpón | P | 500 |
| Hamlet | Teatro | new DateTime(2021, 11, 9, 19, 15, 0) | El Galpón | C 18 | 500 |
| Red Hot Chilli Peppers World Tour | Concierto | new DateTime(2021, 11, 18, 21, 0, 0) | Centenario | C 16 | 500 |
| Eliminatorias Uruguay vs Colombia | Deporte | new DateTime(2021, 10, 7, 19, 45, 0) | Centenario | P | 500 |
| Feria de foodtrucks | Feria Gastronómica | new DateTime(2021, 12, 5, 16, 30, 0) | La Plaza | P | 500 |

## 2.6 TABLA DE DATOS DE COMPRAS

| ID | ActividadNombre | CantidadEntradas | PrecioFinal | usuarioNombre | usuarioApellido | usuarioMail |
|---|---|---|---|---|---|---|
| 1 | Fiesta de la Cerveza | 3 | 500 | Fernando | Fernandez | fernanfernan1@skynet.net |
| 2 | Juntada en la Intendencia | 3 | 500 | Fernando | Fernandez | fefefer@skynet.net |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | Competencia de cartas yugi | 3 | 500 | Es | Talin | comunista104@pueblo.org |
| 4 | Convención del Latu | 3 | 500 | Fernando | Fernandez | fernanfernan1@skynet.net |
| 5 | Paseo por el Castillo | 3 | 500 | Pepe | Mujica | elpepe@presidencia.uy |

# 3 CODIGO FUENTE

## 3.1 SISTEMA

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

namespace Obligatorio1_Dominio
{
   public class Sistema
   {
      #region Listas
      //*************** LISTAS PARA MANEJAR LOS OBJS DEL SISTEMA ***************
      private List<UsuarioRegistrado> usuarios = new List<UsuarioRegistrado>();
      private List<Lugar> lugares = new List<Lugar>();
      private List<Categoria> categorias = new List<Categoria>();
      private List<Actividad> actividades = new List<Actividad>();
      private List<Compra> compras = new List<Compra>();
      #endregion

      #region Pre-Carga de Datos
```

```
    #region Método Constructor de Sistema
    public Sistema() //Metodo constructor de Object
    {
       //Llamada a los métodos que precargan objetos
       PreCargarDatosUsuarios();
       PreCargarDatosLugares();
       PreCargarDatosCategorias();
       PreCargarDatosActividades();
       PreCargarDatosCompras();
    }
    #endregion

    #region Método PreCargarDatosUsuarios
    //*************** CARGA DE DATOS DE USUARIOS HARCODEADOS ***************
    private void PreCargarDatosUsuarios()
    {
       AltaUsuarios("Fernando", "Fernandez", "fernanfernan1@skynet.net", DateTime.Now);
       AltaUsuarios("Robin", "Hood", "comida@paralaplebe.org", DateTime.Now);
       AltaUsuarios("Es", "Talin", "comunista104@pueblo.org", DateTime.Now);
       AltaUsuarios("Fernando", "Fernandez", "fefefer@skynet.net", DateTime.Now);
       AltaUsuarios("Pepe", "Mujica", "elpepe@presidencia.uy", DateTime.Now);
    }
    #endregion

    #region Método PreCargarDatosLugares
    //*************** CARGA DE DATOS DE LUGARES HARCODEADOS ***************
    private void PreCargarDatosLugares()
    {
       AltaLugarAbierto(1, "Plaza Césamo", 400, 500);
       AltaLugarAbierto(2, "El Parque", 400, 500);
```

```
        AltaLugarAbierto(3, "La Plaza", 400, 500);
        AltaLugarAbierto(4, "Centenario", 400, 500);

        AltaLugarCerrado(5, "Intendencia", 400, 500, true, 700, 999);
        AltaLugarCerrado(6, "Castillo Pittamiglio", 400, 500, false, 700, 999);
        AltaLugarCerrado(7, "Latu", 400, 500, true, 700, 999);
        AltaLugarCerrado(8, "El Galpón", 400, 500, true, 700, 999);

    }
    #endregion

    #region Método PreCargarDatosCategorias
    //*************** CARGA DE DATOS DE CATEGORIAS HARCODEADOS ***************
    private void PreCargarDatosCategorias()
    {
        AltaCategoria("Fiesta", "Eventos bailables para borrachines");
        AltaCategoria("Deporte", "Evento de actividad física");
        AltaCategoria("Competencia", "Evento en el que se disputa un premio");
        AltaCategoria("Educación", "Evento de culturización");
        AltaCategoria("Cine", "Proyeccion de películas");
        AltaCategoria("Teatro", "Representación de obras");
        AltaCategoria("Concierto", "Evento musical en vivo");
        AltaCategoria("Feria Gastronómica", "Mercado de alimentos y/o bebidas");

    }
    #endregion

    #region Método PreCargarDatosActividades
    //*************** CARGA DE DATOS DE ACTIVIDADES HARCODEADOS ***************
    private void PreCargarDatosActividades()
```

```
    {
        AltaActividad("Fiesta de la Cerveza", "Fiesta", new DateTime(2021, 10, 5, 20, 0, 0), "Plaza Césamo", "C18", 500);
        AltaActividad("Juntada en la Intendencia", "Deporte", new DateTime(2022, 1, 24, 16, 30, 0), "Intendencia", "C16", 500);
        AltaActividad("Competencia de cartas yugi", "Competencia", new DateTime(2021, 11, 15, 12, 30, 0), "El Parque", "C13", 500);
        AltaActividad("Convención del Latu", "Educación", new DateTime(2021, 10, 9, 16, 30, 0), "Latu", "P", 500);
        AltaActividad("Paseo por el Castillo", "Educación", new DateTime(2021, 12, 15, 10, 30, 0), "Castillo Pittamiglio", "C13", 500);
        AltaActividad("Matrix Resurrections", "Cine", new DateTime(2021, 10, 10, 18, 45, 0), "El Galpón", "P", 500);
        AltaActividad("Hamlet", "Teatro", new DateTime(2021, 11, 9, 19, 15, 0), "El Galpón", "C18", 500);
        AltaActividad("Red Hot Chilli Peppers World Tour", "Concierto", new DateTime(2021, 11, 18, 21, 0, 0), "Centenario", "C16", 500);
        AltaActividad("Eliminatorias Uruguay vs Colombia", "Deporte", new DateTime(2021, 10, 7, 19, 45, 0), "Centenario", "P", 500);
        AltaActividad("Feria de foodtrucks", "Feria Gastronómica", new DateTime(2021, 12, 5, 16, 30, 0), "La Plaza", "P", 500);

    }
    #endregion

    #region Método PreCargarDatosCompras
    //*************** CARGA DE DATOS DE COMPRAS HARCODEADOS ***************
    private void PreCargarDatosCompras()
    {
        AltaCompra(1, "Fiesta de la Cerveza", 3, 500, "Fernando", "Fernandez", "fernanfernan1@skynet.net");
        AltaCompra(2, "Juntada en la Intendencia", 3, 500, "Fernando", "Fernandez", "fefefer@skynet.net");
        AltaCompra(3, "Competencia de cartas yugi", 3, 500, "Es", "Talin", "comunista104@pueblo.org");
        AltaCompra(4, "Convención del Latu", 3, 500, "Fernando", "Fernandez", "fernanfernan1@skynet.net");
        AltaCompra(5, "Paseo por el Castillo", 3, 500, "Pepe", "Mujica", "elpepe@presidencia.uy");
```

```
    }
    #endregion

    #endregion

    #region Método ListarActividades
    //*************** METODO PARA MOSTRAR O LISTAR LAS ACTIVIDADES ***************
    public string ListarActividades()
    {
       string resultadoActividades = "";

       foreach (Actividad miActividad in actividades)
       {
          resultadoActividades += miActividad + "\n";
       }
       return resultadoActividades;
    }
    #endregion

    #region Método ListarCategorias
    //*************** METODO PARA MOSTRAR O LISTAR LAS ACTIVIDADES ***************
    public string ListarCategorias()
    {
       string resultadoCategorias = "";

       foreach (Categoria miCategoria in categorias)
       {
          resultadoCategorias += miCategoria.Nombre + " - ";
       }
       return resultadoCategorias;
```

```
    }
    #endregion

    #region Método CambiarAforoMaximo
    //*************** METODO PARA CAMBIAR EL AFORO MAXIMO ***************
    public string CambiarAforoMaximo(float nuevoPorcentajeAforoMaximo)
    {
      string nuevoValor = "";
      if (nuevoPorcentajeAforoMaximo > 0 && nuevoPorcentajeAforoMaximo < 101)
      {
        LugarCerrado.PorcentajeAforoMaximo = nuevoPorcentajeAforoMaximo;
        nuevoValor = "El nuevo valor para el Aforo Máximo es " + LugarCerrado.PorcentajeAforoMaximo +
"%";
      }
      return nuevoValor;
    }
    #endregion

    #region Método CambiarPrecioButaca
    //*************** CAMBIAR VALOR DEL PRECIO DE BUTACAS QUE SE UTILIZAN EN LUGARES ABIERTOS
***************
    public string CambiarPrecioButaca(float nuevoPrecioButaca)
    {
      string nuevoValor = "";
      if (nuevoPrecioButaca > 0)
      {
        LugarAbierto.PrecioButaca = nuevoPrecioButaca;
        nuevoValor = "El nuevo precio para las Butaca es $" + LugarAbierto.PrecioButaca;
      }
      return nuevoValor;
    }
```

```
    #endregion


    #region Método ActividadesPorCategoriaEntreFechas
    //*************** DADA UNA CATEGORÍA, LISTAR LAS ACTIVIDADES DE ESA CATEGORÍA QUE SE REALICEN EN UN RANGO DE FECHAS DADO ***************
    public string ActividadesPorCategoriaEntreFechas(string categoria, DateTime fechaDesde, DateTime fechaHasta)
    {
      string resultadoActividades = "";
      Categoria miCategoria = BuscarCategoria(categoria);


      if (miCategoria != null && fechaDesde > DateTime.MinValue && fechaHasta > DateTime.MinValue)
      {
        //para evaluar las horas correctamente
        fechaHasta = fechaHasta.AddHours(23); //Agregamos 23 horas a fechaHasta
        fechaHasta = fechaHasta.AddMinutes(59); //Agregamos 59 minutos a fechaHasta
        fechaHasta = fechaHasta.AddSeconds(59); //Agregamos 59 segundos a fechaHasta


        foreach (Actividad miActividad in actividades)
        {
          if (miActividad.Categoria == miCategoria && miActividad.FechaYHora >= fechaDesde && miActividad.FechaYHora <= fechaHasta)
          {
            resultadoActividades += miActividad + "\n";
          }
        }
      }
      return resultadoActividades;
    }
    #endregion
```

```
#region Método EspectaculosATP
//*************** LISTAR ESPECTÁCULOS APTOS PARA TODO PÚBLICO ***************
public string EspectaculosATP()
{
   string resultadoEspectaculos = "";
   foreach (Actividad miActividad in actividades)
   {
      if (miActividad.EdadMinima == "P")
      {
         resultadoEspectaculos += miActividad + "\n";
      }
   }
   return resultadoEspectaculos;
}
#endregion

#region Método TieneEdadSuficiente
//*************** METODO QUE VERIFICA EDAD MINIMA ***************
public bool TieneEdadSuficiente(string edadActividad, UsuarioRegistrado usuarioInteresado)
{
   bool resultado = true; //variable de retorno, true por defecto
   // Cálculo de edad del usuario
   int edadUsuario = DateTime.Now.Year - usuarioInteresado.FechaNacimiento.Year;

   // Posibles valores de edadActividad: P, C13, C16, C18
   // Cálculo de edad suficiente
   switch (edadActividad)
   {
      case "P":
         break; // el resultado de un case "P" siempre será true
```

```
            case "C13":
                if (edadUsuario < 13) { resultado = false; } // si edadComprador no llega a la edad mínima, devolveremos "resultado" con valor false
                break;
            case "C16":
                if (edadUsuario < 16) { resultado = false; }
                break;
            case "C18":
                if (edadUsuario < 18) { resultado = false; }
                break;
            default: resultado = false; // si se ingresa una opción no válida, devolvemos False
                break;
        }
        return resultado; //devolvemos resultado
    }
    #endregion

    #region Método BuscarCategoría
    //*************** METODO PARA BUSCAR UNA CATEGORIA ***************
    private Categoria BuscarCategoria(string nombre)
    {
        Categoria categoriaBuscada = null;
        int i = 0;
        while (i < categorias.Count && categoriaBuscada == null)
        {
            if (categorias[i].Nombre.ToUpper() == nombre.ToUpper())
            {
                categoriaBuscada = categorias[i];
            }
            i++;
```

```
        }
        return categoriaBuscada;
    }
    #endregion

    #region Método BuscarCategoríaBool
    //Este método devolverá un bool para uso en Program
    public bool BuscarCategoriaBool(string nombre)
    {
        if (BuscarCategoria(nombre) != null)
        { return true; }
        else
        { return false; }
    }
    #endregion

    #region Método BuscarLugar
    //*************** METODO PARA BUSCAR UN LUGAR ***************
    private Lugar BuscarLugar(string nombre)
    {
        Lugar lugarBuscado = null;
        int i = 0;
        while (i < lugares.Count && lugarBuscado == null)
        {
            if (lugares[i].Nombre.ToUpper() == nombre.ToUpper())
            {
                lugarBuscado = lugares[i];
            }
            i++;
        }
```

```
        return lugarBuscado;
    }
    #endregion

    #region Método BuscarActividad
    //*************** METODO PARA BUSCAR UNA ACTIVIDAD ***************
    private Actividad BuscarActividad(string nombre)
    {
        Actividad actividadBuscada = null;
        int i = 0;
        while (i < actividades.Count && actividadBuscada == null)
        {
            if (actividades[i].Nombre.ToUpper() == nombre.ToUpper())
            {
                actividadBuscada = actividades[i];
            }
            i++;
        }
        return actividadBuscada;
    }
    #endregion

    #region Método BuscarUsuario
    //*************** METODO PARA BUSCAR UN USUARIO ***************
    private UsuarioRegistrado BuscarUsuario(string nombre, string apellido, string mail)
    {
        UsuarioRegistrado usuarioBuscado = null; //usuarioBuscado tiene por defecto valor null
        int i = 0; //contador
        while (i < usuarios.Count && usuarioBuscado == null) //loop de la lista usuarios, mientras
usuarioBuscado valga null
```

```
        {
            //NOTA: Como clave primaria, usaremos: nombre, apellido y mail (de esta forma, podremos tener usuarios con el mismo nombre y apellido (pensando en los nombres más comunes por ej Juan Pérez, Fulano Fernandez, etc).
            if (usuarios[i].Nombre.ToUpper() == nombre.ToUpper() && usuarios[i].Apellido.ToUpper() == apellido.ToUpper() && usuarios[i].Mail.ToUpper() == mail.ToUpper()) //Si algún usuario tiene la misma combinación de nombre, apellido y mail
            {
                usuarioBuscado = usuarios[i]; //devolver ese usuario
            }
            i++; //incrementar contador
        }
        return usuarioBuscado; //devolver valor de usuarioBuscado
    }

    #endregion

    #region BuscarCompra
    //*************** METODO PARA BUSCAR UNA COMPRA ***************
    private Compra BuscarCompra(int id)
    {
        Compra compraBuscado = null;
        int i = 0;
        while (i < compras.Count && compraBuscado == null)
        {
            if (compras[i].Id == id)
            {
                compraBuscado = compras[i];
            }
            i++;
        }
```

```
        return compraBuscado;
    }
    #endregion

    #region Método AltaLugarAbierto
    //*************** METODO PARA DAR DE ALTA UN LUGAR ABIERTO***************
    private void AltaLugarAbierto(int id, string nombre, float dimensionesLargo, float dimensionesAncho)
    {
        if (nombre != "" && dimensionesAncho > 0 && dimensionesLargo > 0 && id > 0) //si los datos no están
vacíos
        {
            if (BuscarLugar(nombre) == null)
            {
                this.lugares.Add(new LugarAbierto(id, nombre, dimensionesLargo, dimensionesAncho));
            }
        }
    }
    #endregion

    #region Método AltaLugarCerrado
    //*************** METODO PARA DAR DE ALTA UN LUGAR CERRADO***************
    private void AltaLugarCerrado(int id, string nombre, float dimensionesAncho, float dimensionesLargo, bool
accesibilidad, float valorMantenimiento, float precioCerrado)
    {
        if (nombre != "" && dimensionesAncho > 0 && dimensionesLargo > 0 && id > 0 && valorMantenimiento
> 0 && precioCerrado > 0) //si los datos no están vacíos
        {
            if (BuscarLugar(nombre) == null)
            {
                this.lugares.Add(new LugarCerrado(id, nombre, dimensionesAncho, dimensionesLargo,
accesibilidad, valorMantenimiento, precioCerrado));
```

```
            }
        }
    }
    #endregion

    #region Método AltaCategoria
    //*************** METODO PARA DAR DE ALTA UNA CATEGORIA***************
    private void AltaCategoria(string nombre, string descripcion)
    {
        if (nombre != "" && descripcion != "") //si los datos no están vacíos
        {
            if (BuscarCategoria(nombre) == null)
            {
                this.categorias.Add(new Categoria(nombre, descripcion));
            }
        }
    }
    #endregion

    #region Método AltaActividad
    //*************** METODO PARA DAR DE ALTA UNA ACTIVIDAD***************
    private void AltaActividad(string nombre, string miCategoria, DateTime fechaYHora, string miLugar, string
edadMinima, float precioEntrada)
    {
        Lugar lugarObj = BuscarLugar(miLugar);
        Categoria categoriaObj = BuscarCategoria(miCategoria);
        if (nombre != "" && categoriaObj != null && fechaYHora > DateTime.MinValue && lugarObj != null &&
ValidarCodigoDeEdad(edadMinima) && precioEntrada > 0) //si los datos no están vacíos
        {
            this.actividades.Add(new Actividad(nombre, categoriaObj, fechaYHora, lugarObj, edadMinima,
precioEntrada));
```

```
        }
    }
    #endregion

    #region Método Auxiliar: ValidarCodigoDeEdad
    //*************** METODO AUXILIAR PARA VALIDAR EL CODIGO DE EDAD MINIMA***************
    private bool ValidarCodigoDeEdad(string texto)
    {
        if (texto != "P" && texto != "C13" && texto != "C16" && texto != "C18")
        {
            return false;
        }
        else
        {
            return true;
        }
    }
    #endregion

    #region Método AltaUsuarios
    //*************** METODO PARA DAR DE ALTA UN USUARIO***************
    private void AltaUsuarios(string nombre, string apellido, string mail, DateTime fechaNacimiento)
    {
        if (nombre != "" && apellido != "" && mail != "" && fechaNacimiento > DateTime.MinValue) //si los datos no están vacíos
        {
            if (BuscarUsuario(nombre, apellido, mail) == null) //confirmamos que el usuario no exista todavía
            {
                this.usuarios.Add(new UsuarioRegistrado(nombre, apellido, mail, fechaNacimiento)); //agregamos una nueva instancia de UsuarioRegistrado a la lista de usuarios
            }
```

```
        }
    }
    #endregion

    #region Método AltaCompra
    //*************** METODO PARA DAR DE ALTA UNA COMPRA***************
    private void AltaCompra(int id, string actividadNombre, int cantidadEntradas, float precioFinal, string usuarioNombre, string usuarioApellido, string usuarioMail)
    {
        Actividad actividadObj = BuscarActividad(actividadNombre);
        UsuarioRegistrado usuarioObj = BuscarUsuario(usuarioNombre, usuarioApellido, usuarioMail);
        bool puedeEntrar = TieneEdadSuficiente(actividadObj.EdadMinima, usuarioObj);
        if (id > 0 && actividadObj != null && cantidadEntradas > 0 && precioFinal > 0 && usuarioObj != null && puedeEntrar) //verificamos los datos (que no estén vacíos, que los números sean > 0 y que la edad del usuario sea la apropiada)
        {
            if (BuscarCompra(id) == null)
            {
                this.compras.Add(new Compra(id, actividadObj, cantidadEntradas, precioFinal, usuarioObj));
            }
        }
    }
    #endregion

  }
}
```

## 3.2 USUARIO REGISTRADO

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

```
namespace Obligatorio1_Dominio
{
   public class UsuarioRegistrado
   {
      #region Atributos
      private int id;
      private static int ultimoId = 1;
      private string nombre;
      private string apellido;
      private string mail;
      private DateTime fechaNacimiento;
      #endregion

      #region Propiedades
      public int Id
      {
         get { return this.id; }
      }
      public string Nombre
      {
         get { return this.nombre; }
      }
      public string Apellido
      {
         get { return this.apellido; }
      }
      public string Mail
      {
         get { return this.mail; }
```

```
        }
        public DateTime FechaNacimiento
        {
            get { return this.fechaNacimiento; }
        }
        #endregion

        #region Constructores
        //*************** METODO CONSTRUCTOR DE USUARIO***************
        public UsuarioRegistrado(string nombre, string apellido, string mail, DateTime fechaNacimiento)
        {
            this.id = UsuarioRegistrado.ultimoId; //id autonumérico de instancia
            UsuarioRegistrado.ultimoId++; //+1 al contador de instancias de la clase UsuarioRegistrado
            this.nombre = nombre;
            this.apellido = apellido;
            this.mail = mail;
            this.fechaNacimiento = fechaNacimiento;
        }
        #endregion
    }
}
```

## 3.3 ACTIVIDAD

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

namespace Obligatorio1_Dominio
```

```
{
    public class Actividad
    {
        #region Atributos
        private int id;
        private static int ultimoId = 1;
        private string nombre;
        private Categoria categoria;
        private DateTime fechaYHora;
        private Lugar lugar;
        private string edadMinima;
        private float precioEntrada;
        private int meGusta;
        #endregion

        #region Propiedades
        public int Id
        {
            get { return this.id; }
        }
        public string Nombre
        {
            get { return this.nombre; }
        }
        public Categoria Categoria
        {
            get { return this.categoria; }
        }
        public DateTime FechaYHora
        {
```

```
        get { return this.fechaYHora; }
    }

    public Lugar Lugar
    {
        get { return this.lugar; }
    }
    public string EdadMinima
    {
        get { return this.edadMinima; }
    }
    public float PrecioEntrada
    {
        get { return this.precioEntrada; }
        set { this.precioEntrada = value; }
    }
    public int MeGusta
    {
        get { return this.meGusta; }
        //set { this.meGusta = value; } //para el futuro, cuando el valor de meGusta deba actualizarse
    }
    #endregion

    #region Constructores
    //*************** METODO CONSTRUCTOR DE ACTIVIDAD***************
    public Actividad(string nombre, Categoria miCategoria, DateTime fechaYHora, Lugar miLugar, string
edadMinima, float precioEntrada)
    {
        this.id = Actividad.ultimoId; //asignamos el valor del atrib de clase "ultimoId" como atrib "id" de la
instancia
        Actividad.ultimoId++; //aumentamos el atributo contador de clase, ultimoId
```

```
            this.nombre = nombre;
            this.categoria = miCategoria;
            this.fechaYHora = fechaYHora;
            this.lugar = miLugar;
            this.edadMinima = edadMinima;
            this.precioEntrada = precioEntrada;
            this.meGusta = 0; //Contador miembro de instancia que incrementará luego de creado el obj Actividad, cuando se implemente una función para aumentar los "Me Gusta"
        }
        #endregion

        #region ToString Actividad
        public override string ToString()
        {
            return $"\nNombre: {this.nombre}\nCategoría: {this.categoria}\nFecha: {this.fechaYHora}\nLugar: {this.lugar}\nEdad mínima: {this.edadMinima}"; //\nPrecio: ${this.precioEntrada}
        }
        #endregion
    }
}
```

## 3.4 COMPRA

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

namespace Obligatorio1_Dominio
{
    public class Compra
    {
```

```
    #region Atributos
    private int id;
    private Actividad actividad;
    private int cantidadEntradas;
    private DateTime fechaYHora;
    private bool estaActiva;
    private float precioFinal;
    private UsuarioRegistrado usuarioComprador;
    #endregion

    #region Constructores
    //*************** METODO CONSTRUCTOR DE COMPRA***************
    public Compra(int id, Actividad actividad, int cantidadEntradas, float precioFinal, UsuarioRegistrado
usuarioComprador)
    {
       this.id = id;
       this.actividad = actividad;
       this.cantidadEntradas = cantidadEntradas;
       this.fechaYHora = DateTime.Now; //La compra se efectúa en el momento en que se da de alta un objeto
tipo Compra
       this.estaActiva = true; //Por defecto, al crear la instancia, el atributo estaActiva será true, dado que la
compra acaba de ocurrir
       this.precioFinal = precioFinal;
       this.usuarioComprador = usuarioComprador;
    }
    #endregion

    #region Propiedades
    public int Id
    {
       get { return this.id; }
```

```
        }
        public Actividad Actividad
        {
            get { return this.actividad; }
            set { this.actividad = value; } //para actualizar si la compra se canceló o no
        }
        public int CantidadEntradas
        {
            get { return this.cantidadEntradas; }
        }
        public DateTime FechaYHora
        {
            get { return this.fechaYHora; }
        }
        public bool EstaActiva
        {
            get { return this.estaActiva; }
        }
        public float PrecioFinal
        {
            get { return this.precioFinal; }
        }
        public UsuarioRegistrado UsuarioComprador
        {
            get { return this.usuarioComprador; }
        }
        #endregion
    }
}
```

## 3.5 CATEGORIA

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

namespace Obligatorio1_Dominio
{
   public class Categoria
   {
      #region Atributos
      private string nombre;
      private string descripcion;
      #endregion

      #region Propiedades
      public string Nombre
      {
         get { return this.nombre; }
      }
      public string Descripcion
      {
         get { return this.descripcion; }
      }
      #endregion

      #region Constructores
      //*************** METODO CONSTRUCTOR DE CATEGORIA***************
      public Categoria(string nombre, string descripcion)
      {
```

```
            this.nombre = nombre;
            this.descripcion = descripcion;
        }
        #endregion

        #region ToString Categoria
        public override string ToString()
        {
            return $"{this.nombre} ({this.descripcion})";
        }
        #endregion
    }
}
```

## 3.6 LUGAR

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

namespace Obligatorio1_Dominio
{
    public abstract class Lugar
    {
        #region Atributos
        private int id;
        private string nombre;
        private float dimensionesAncho;
        private float dimensionesLargo;
        #endregion
```

```
#region Propiedades
public int Id
{
   get { return this.id; }
}
public string Nombre
{
   get { return this.nombre; }
}
public float DimensionesAncho
{
   get { return this.dimensionesAncho; }
}
public float DimensionesLargo
{
   get { return this.dimensionesLargo; }
}

#endregion

#region Constructores
//*************** METODO CONSTRUCTOR DE LUGAR***************
public Lugar(int id, string nombre, float dimensionesLargo, float dimensionesAncho)
{
   this.id = id;
   this.nombre = nombre;
   this.dimensionesLargo = dimensionesLargo;
   this.dimensionesAncho = dimensionesAncho;
}
```

```
        #endregion


        #region ToString Actividad
        public override string ToString()
        {
            return this.nombre;
        }
        #endregion
    }
}
```

## 3.7 LUGAR ABIERTO

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

namespace Obligatorio1_Dominio
{

    public class LugarAbierto : Lugar
    {
        #region Atributos
        private static float precioButaca = 150;
        #endregion


        #region Constructores
        //*************** METODO CONSTRUCTOR DE LUGAR ABIERTO***************
        public LugarAbierto(int id, string nombre, float dimensionesLargo, float dimensionesAncho) : base(id,
nombre, dimensionesLargo, dimensionesAncho)
        {
```

```
        }
        #endregion

        #region Propiedades
        public static float PrecioButaca
        {
            set { LugarAbierto.precioButaca = value; }
            get { return LugarAbierto.precioButaca; }
        }
        #endregion
    }
}
```

## 3.8 LUGAR CERRADO

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

namespace Obligatorio1_Dominio
{
    public class LugarCerrado : Lugar
    {

        #region Atributos
        private bool accesibilidad;
        private float valorMantenimiento;
        private float precioCerrado;
        private static float porcentajeAforoMaximo = 43;
        #endregion
```

```
        #region Constructores

        //*************** METODO CONSTRUCTOR DE LUGAR CERRADO***************

        public LugarCerrado(int id, string nombre, float dimensionesLargo, float dimensionesAncho, bool
accesibilidad, float valorMantenimiento, float precioCerrado) : base(id, nombre, dimensionesLargo,
dimensionesAncho)

        {

            this.accesibilidad = accesibilidad;

            this.valorMantenimiento = valorMantenimiento;

            this.precioCerrado = precioCerrado;

        }

        #endregion


        #region Propiedades

        public static float PorcentajeAforoMaximo

        {

            set { LugarCerrado.porcentajeAforoMaximo = value; }

            get { return LugarCerrado.porcentajeAforoMaximo; }

        }

        #endregion

    }

}
```