SANTIAGO CAMPS - 219114
Programación 2 - Grupo M2C - Liliana Pino

Datos precargados

Administrador:

admin@eventos17.com
Admin!99

Organizadores:

org@eventos17.com
Org!9999
Carlos
099542788
Calle 123

lucas@mail.com
Lucas!111
Lucas
0996556545
Avenida 456

Eventos:

TIPO: Evento Común
FECHA: 2018, 10, 01
TURNO: tarde
DESCRIPCIÓN: Cumpleaños de mi tío
NOMBRE CLIENTE: Eduardo Gonzalez
SERVICIOS: Parrilla 9, Cabalgata 7
ASISTENTES: 9
FOTO: evento01.jpg
DURACIÓN: 3
ORGANIZADOR: org@eventos17.com

TIPO: Evento Premium
FECHA: 2018, 03, 12
TURNO: noche
DESCRIPCIÓN: Aniversario del club.
NOMBRE CLIENTE: Alberto P.
SERVICIOS: Comida 75, Parrilla 25
ASISTENTES: 100
FOTO: evento02.jpg
ORGANIZADOR: org@eventos17.com

TIPO: Evento Común
FECHA: 2018, 05, 30
TURNO: mañana
DESCRIPCIÓN: Reunión de amigos.
NOMBRE CLIENTE: Eduardo Gonzalez
SERVICIOS: Cabalgata 7
ASISTENTES: 9
FOTO: evento05.jpg
DURACIÓN: 4
ORGANIZADOR: org@eventos17.com

TIPO: Evento Común
FECHA: 2018, 07, 27
TURNO: tarde
DESCRIPCIÓN: Reunión de negocios.
NOMBRE CLIENTE: Fernando P.
SERVICIOS: Comida 7
ASISTENTES: 7
FOTO: evento03.jpeg
DURACIÓN: 4
ORGANIZADOR: lucas@mail.com

TIPO: Evento Premium
FECHA: 2019, 04, 08
TURNO: noche
DESCRIPCIÓN: Cumpleaños de Roberto.
NOMBRE CLIENTE: Alberto P.
SERVICIOS: Parrilla 75, Cabalgata 33, Cotillon 99
ASISTENTES: 100
FOTO: evento04.jpg
ORGANIZADOR: lucas@mail.com

Código comentado

```
public class Empresa
    {
        private List<Usuario> usuarios = new List<Usuario>();
        private List<Servicio> servicios = new List<Servicio>();
        private List<Evento> eventos = new List<Evento>();
        private static Empresa instancia;

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

        public List<Servicio> Servicios
        {
            get { return servicios; }
        }


        // Constructor
        public Empresa()
        {
            PrecargarDatos();
        }

        // Precargar Datos
        private void PrecargarDatos()
        {
            AltaServicio("Cabalgata", "Un recorrido a caballo por un maravilloso paisaje.",
200);
            AltaServicio("Comida", "Exelentes platos de comida muy rica.", 400);
            AltaServicio("Parrilla", "Variedad de exquisitas carnes a la parrilla.", 550);
            AltaServicio("Cotillon", "Chucherias para una fiesta bien divertida.", 120);

            AltaAdministrador("admin@eventos17.com", "Admin!99");

            AltaOrganizador("org@eventos17.com", "Org!9999", "Carlos", "099542788", "Calle
123");
            AltaOrganizador("lucas@mail.com", "Lucas!111", "Lucas", "0996556545", "Avenida
456");

            AltaEventoComun(new DateTime(2018, 10, 01), "tarde", "Cumpleaños de mi tío.",
"Eduardo Gonzalez", "Parrilla", 9, 9, "evento01.jpg", 3);
            AgregarServicioEvento("Cabalgata", 7, new DateTime(2018, 10, 01));
            AgregarEventoOrganizador(new DateTime(2018, 10, 01), "org@eventos17.com");

            AltaEventoPremium(new DateTime(2018, 03, 12), "noche", "Aniversario del club.",
"Alberto P.", "Comida", 75, 100, "evento02.jpg");
            AgregarServicioEvento("Parrilla", 25, new DateTime(2018, 03, 12));
            AgregarEventoOrganizador(new DateTime(2018, 03, 12), "org@eventos17.com");
```

```
            AltaEventoComun(new DateTime(2018, 05, 30), "mañana", "Reunión de amigos.",
"Eduardo Gonzalez", "Cabalgata", 7, 9, "evento05.jpg", 4);
            AgregarEventoOrganizador(new DateTime(2018, 05, 30), "org@eventos17.com");

            AltaEventoComun(new DateTime(2018, 07, 27), "tarde", "Reunión de negocios.",
"Fernando P.", "Comida", 7, 7, "evento03.jpeg", 4);
            AgregarEventoOrganizador(new DateTime(2018, 07, 27), "lucas@mail.com");

            AltaEventoPremium(new DateTime(2019, 04, 08), "noche", "Cumpleaños de Roberto.",
"Alberto P.", "Parrilla", 75, 100, "evento04.jpg");
            AgregarServicioEvento("Cabalgata", 33, new DateTime(2019, 04, 08));
            AgregarServicioEvento("Cotillon", 99, new DateTime(2019, 04, 08));
            AgregarEventoOrganizador(new DateTime(2019, 04, 08), "lucas@mail.com");
        }



        // Buscar Servicio
        public Servicio BuscarServicio(string nombre)
        {
            // Busca un servicio según su nombre y lo devuelve, si no existe devuelve null
            Servicio servicio = null;
            int i = 0;
            while (i < servicios.Count && servicio == null)
            {
                if (servicios[i].Nombre == nombre)
                {
                    servicio = servicios[i];
                }
                i++;
            }
            return servicio;
        }

        // Alta Servicio
        public bool AltaServicio(string nombre, string descripcion, decimal precioPorPersona)
        {
            bool alta = false;
            Servicio servicio = BuscarServicio(nombre);
            int cantServicios = servicios.Count;
            if (servicio == null)
            {
                servicios.Add(new Servicio(nombre, descripcion, precioPorPersona));
            }
            if (cantServicios < servicios.Count)
            {
                alta = true;
            }
            return alta;
        }

        // Listar Servicios
        public string ListarServicios()
        {
            // Lista todos los servicios existentes
            string lista = "";
            foreach (Servicio servicio in servicios)
            {
                lista += servicio.ToString() + "\n";
            }
            return lista;
```

```
        }



        // Buscar Usuario
        public Usuario BuscarUsuario(string mail)
        {
            // Busca un usuario según su mail y lo devuelve, null si no está registrado
            Usuario usuario = null;
            int i = 0;
            while (i<usuarios.Count && usuario == null)
            {
                if (usuarios[i].Mail == mail)
                {
                    usuario = usuarios[i];
                }
                i++;
            }
            return usuario;
        }

        // Validar Usuario y su contrasenia
        public string ValidarUsuario(string mail, string contrasenia)
        {
            string rol = "";
            Usuario usuario = BuscarUsuario(mail);
            if (usuario != null && usuario.Contrasenia == contrasenia)
            {
                rol = usuario.MiRol();
            }
            return rol;
        }


        // Alta Administrador
        public bool AltaAdministrador(string mail, string contrasenia)
        {
            bool alta = false;
            Usuario usuario = BuscarUsuario(mail);
            int cantUsuarios = usuarios.Count;
            bool datosValidados = ValidarMail(mail) && ValidarContrasenia(contrasenia);
            if (usuario == null && datosValidados)
            {
                usuarios.Add(new Usuario(mail, contrasenia));
            }
            if (cantUsuarios < usuarios.Count)
            {
                alta = true;
            }
            return alta;
        }

        // Alta Organizador
        public bool AltaOrganizador(string mail, string contrasenia, string nombre, string
telefono, string direccion)
        {
            bool alta = false;
            Usuario usuario = BuscarUsuario(mail);
            int cantUsuarios = usuarios.Count;
```

```
            bool datosValidados = ValidarMail(mail) && ValidarContrasenia(contrasenia) &&
ValidarNombre(nombre);
            if (usuario == null && datosValidados)
            {
                usuarios.Add(new Organizador(mail, contrasenia, nombre, telefono, direccion));
            }
            if (cantUsuarios < usuarios.Count)
            {
                alta = true;
            }
            return alta;
        }

        // Validar si Usuario es Organizador
        public bool UsuarioEsOrganizador(string mail)
        {
            // Devuelve true si el usuario es de la clase Organizador
            bool validado = false;
            Usuario usuario = BuscarUsuario(mail);
            if (usuario != null && usuario.MiRol() == "Organizador")
            {
                validado = true;
            }
            return validado;
        }

        // Datos de Organizador
        public string DatosDeOrganizador(string mail)
        {
            // Devuelve los datos del Organizador
            string datos = "";
            Usuario usuario = BuscarUsuario(mail);
            Organizador organizador = null;
            if (usuario != null && UsuarioEsOrganizador(mail))
            {
                organizador = (Organizador)usuario;
                datos = organizador.MisDatos();
            }
            return datos;
        }

        // Agregar Evento a Organizador
        public bool AgregarEventoOrganizador(DateTime fechaEvento, string mailOrganizador)
        {
            // Agrega un Evento a la lista de Eventos de un Organizador y devuelve true, si el
usuario es organizador y si el evento existe
            bool agregado = false;
            Evento evento = BuscarEvento(fechaEvento);
            Usuario usuario = BuscarUsuario(mailOrganizador);
            Organizador organizador = null;
            if (usuario != null && UsuarioEsOrganizador(mailOrganizador))
            {
                organizador = (Organizador)usuario;
            }
            if (evento != null && organizador != null)
            {
                agregado = organizador.AgregarEvento(evento);
            }
            return agregado;
        }
```

```
// Listar Usuarios Todos
public string ListarUsuariosTodos()
{
    // Devuelve una lista de todos los usuario con sus datos y su rol
    string lista = "";
    foreach ( Usuario usuario in usuarios)
    {
        lista += "ROL: " + usuario.MiRol() + " " + usuario.ToString() + "\n";
    }
    return lista;
}

// Listar Organizadores
public List<Organizador> ListarOrganizadores()
{
    List<Organizador> lista = new List<Organizador>();
    foreach (Usuario usuario in usuarios)
    {
        if (usuario.MiRol() == "Organizador")
        {
            Organizador organizador = (Organizador)usuario;
            lista.Add(organizador);
        }
    }

    return lista;
}

// Listar Administradores
public List<Usuario> ListarAdministradores()
{
    List<Usuario> lista = new List<Usuario>();
    foreach (Usuario usuario in usuarios)
    {
        if (usuario.MiRol() == "Administrador")
        {
            lista.Add(usuario);
        }
    }

    return lista;
}



// Buscar Evento
public Evento BuscarEvento(DateTime fecha)
{
    // Busca si un evento existe y lo devuelve, de lo contrario devuelve null
    Evento evento = null;
    int i = 0;
    while (i < eventos.Count && evento == null)
    {
        if (eventos[i].Fecha.Year == fecha.Year &&
            eventos[i].Fecha.Month == fecha.Month &&
            eventos[i].Fecha.Day == fecha.Day)
        {
            evento = eventos[i];
        }
        i++;
    }
```

```
            return evento;
        }

        // Alta Evento Comun
        public bool AltaEventoComun(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes, string
foto, int duracion)
        {
            bool alta = false;
            bool datosValidados = ValidarEventoComun(fecha, turnoString, descripcion,
nombreCliente, nombreServicio, cantPersonasServicio, asistentes, foto, duracion);
            Turno turno = BuscarTurno(turnoString);
            Servicio servicio = BuscarServicio(nombreServicio);
            Evento evento = BuscarEvento(fecha);
            int cantEventos = eventos.Count;
            if (evento == null && datosValidados && turno != 0)
            {
                eventos.Add(new EventoComun(fecha, turno, descripcion, nombreCliente, servicio,
cantPersonasServicio, asistentes, foto, duracion));
            }
            if (cantEventos < eventos.Count)
            {
                alta = true;
            }
            return alta;
        }

        // Alta Evento Premium
        public bool AltaEventoPremium(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes, string
foto)
        {
            bool alta = false;
            bool datosValidados = ValidarEventoPremium(fecha, turnoString, descripcion,
nombreCliente, nombreServicio, cantPersonasServicio, asistentes, foto);
            Turno turno = BuscarTurno(turnoString);
            Servicio servicio = BuscarServicio(nombreServicio);
            Evento evento = BuscarEvento(fecha);
            int cantEventos = eventos.Count;
            if (evento == null && datosValidados && turno != 0)
            {
                eventos.Add(new EventoPremium(fecha, turno, descripcion, nombreCliente,
servicio, cantPersonasServicio, asistentes, foto));
            }
            if (cantEventos < eventos.Count)
            {
                alta = true;
            }
            return alta;
        }

        // Agregar Servicio a Evento
        public bool AgregarServicioEvento(string nombreServicio, int cantPersonas, DateTime
fechaEvento)
        {
            bool agregado = false;
            Servicio servicio = BuscarServicio(nombreServicio);
            Evento evento = BuscarEvento(fechaEvento);
            if (servicio != null &&
                evento != null &&
                cantPersonas <= evento.Asistentes)
```

```
            {
                agregado = evento.AgregarServicioContratado(servicio, cantPersonas);
            }
            return agregado;
        }

        // Ver Porcentaje de Aumento para Eventos Premium
        public int VerPorcentajeDeAumento()
        {
            // Muestra el ese dato
            return EventoPremium.PorcentajeDeAumento;
        }

        // Cambiar Porcentaje de Aumento para Eventos Premium
        public bool CambiarPorcentajeDeAumento(int nuevoValor)
        {
            // Cambia ese dato
            return EventoPremium.CambiarPorcentajeDeAumento(nuevoValor);
        }

        // Ver Monto Fijo de Limpieza para Eventos Comunes
        public decimal VerMontoFijoLimpieza()
        {
            // Muestra ese dato
            return EventoComun.MontoFijoLimpieza;
        }

        // Cambiar Monto Fijo de Limpieza para Eventos Comunes
        public bool CambiarMontoFijoLimpieza(decimal nuevoValor)
        {
            // Cambia ese dato
            return EventoComun.CambiarMontoFijoLimpieza(nuevoValor);
        }

        // Listar Eventos de Organizador STRING
        public string ListarEventosOrganizador(string mail, string contrasenia)
        {
            // Lista todos los Eventos del Organizador si sus datos son correctos, si no es
Organizador o si no existe muestra un aviso
            string lista = "";
            Usuario usuario = BuscarUsuario(mail);
            if (usuario == null || usuario.Contrasenia != contrasenia)
            {
                lista = "El usuario ingresado o la contraseña no son correctos";
            }
            else
            {
                lista = usuario.ListarEventos();
            }
            return lista;
        }

        // Listar Eventos de Organizador LISTA DE EVENTOS
        public List<Evento> ListarEventosOrganizador(string mail)
        {
            // Lista todos los Eventos del Organizador si sus datos son correctos, si no es
Organizador o si no existe muestra un aviso
            List<Evento> lista = null;
            Usuario usuario = BuscarUsuario(mail);
            Organizador organizador = null;
            if (usuario != null && UsuarioEsOrganizador(mail))
```

```
    {
        organizador = (Organizador)usuario;
        lista = organizador.Eventos;
    }
    return lista;
}

// Mostrar datos de Evento
public string MostrarDatosDeEvento(DateTime fecha)
{
    // Muestra los datos de un Evenot, si es que existe
    string datos = "";
    Evento evento = BuscarEvento(fecha);
    if (evento != null)
    {
        datos += evento.ToString() + "\n" + evento.DatosDeMisServicios();
    }
    return datos;
}

// Mostrar datos de Evento
public List<ServicioContratado>ServiciosDeEvento(DateTime fecha)
{
    // Muestra los datos de un Evenot, si es que existe
    List<ServicioContratado> servicios = new List<ServicioContratado>();

    Evento evento = BuscarEvento(fecha);
    if (evento != null)
    {
        servicios=evento.ServiciosContratados();
    }
    return servicios;
}

// Listar Eventos entre dos fechas ordenados por organizador(asc)/fecha(desc)
public List<Evento> EventosEntre(DateTime fecha1, DateTime fecha2)
{
    List<Evento> lista = new List<Evento>();
    if (fecha1 > fecha2) // hago que fecha1 siempre sea la menor
    {
        DateTime fechaAux = fecha1;
        fecha1 = fecha2;
        fecha2 = fechaAux;
    }

    usuarios.Sort(); // Ordeno por mail a los usuarios
    Organizador organizador = null;

    foreach (Usuario usuario in usuarios)
    {
        if (UsuarioEsOrganizador(usuario.Mail))
        {
            organizador = (Organizador)usuario;
            lista.AddRange(organizador.MisEventosEntreFechasOrdenados(fecha1, fecha2));
        }
    }

    return lista;
}

// Total generado por eventos entre dos fechas
```

```
public decimal EventosEntreTotalGenerado(DateTime fecha1, DateTime fecha2)
{
    decimal total = 0;
    if (fecha1 > fecha2) // hago que fecha1 siempre sea la menor
    {
        DateTime fechaAux = fecha1;
        fecha1 = fecha2;
        fecha2 = fechaAux;
    }
    List<Evento> eventosEntre = EventosEntre(fecha1, fecha2);
    foreach (Evento evento in eventosEntre)
    {
        total += evento.CostoCalculado;
    }
    return total;
}

// Listar eventos sin un servicio determinado
public List<Evento> EventosSinServicio(Servicio servicio)
{
    List<Evento> lista = new List<Evento>();
    foreach (Evento evento in eventos)
    {
        if (!evento.TengoEsteServicio(servicio))
        {
            lista.Add(evento);
        }
    }
    return lista;
}

// Ver si evento tiene un servicio
public bool EventoTieneServicio(DateTime fechaEvento, string servicioNombre)
{
    bool tiene = false;
    Evento evento = BuscarEvento(fechaEvento);
    Servicio servicio = BuscarServicio(servicioNombre);
    if (evento != null && servicio != null)
    {
        if (evento.TengoEsteServicio(servicio))
        {
            tiene = true;
        }
    }
    return tiene;
}


// Ver Organizador de Evento
public Organizador VerOrgDeEvento(Evento evento)
{
    // TODO: preguntar si esto está bien
    Organizador org = null;
    Organizador orgCasteado = null;
    int i = 0;
    while (i < usuarios.Count && org == null)
    {
        if (usuarios[i].MiRol() == "Organizador")
        {
            orgCasteado = (Organizador)usuarios[i];
            if (orgCasteado.Eventos.Contains(evento))
```

```
            {
                org = orgCasteado;
            }
        }
        i++;
    }
    return org;
}

// Calcular total generado por Organizador
public decimal TotalGeneradoPorOrganizador(string mail)
{
    decimal total = 0;
    Usuario usuario = BuscarUsuario(mail);
    Organizador organizador = null;
    if (usuario != null && UsuarioEsOrganizador(mail))
    {
        organizador = (Organizador)usuario;
        total = organizador.TotalGeneradoPorMisEventos();
    }
    return total;
}




// Validar Mail
public bool ValidarMail(string mail)
{
    return Usuario.ValidarMail(mail);
}

// Validar Contraseña
public bool ValidarContrasenia(string contrasenia)
{
    return Usuario.ValidarContrasenia(contrasenia);
}

// Validar Nombre (min 3 letras)
public bool ValidarNombre(string nombre)
{
    return Usuario.ValidarNombre(nombre);
}

// Buscar turno
public Turno BuscarTurno(string turnoString)
{
    // Busca un Turno y si existe devuelve ese enum
    Turno turno = 0;
    string[] turnos = Enum.GetNames(typeof(Turno));
    if (turnos.Contains(turnoString))
    {
        turno = (Turno)Enum.Parse(typeof(Turno), turnoString);
    }
    return turno;
}

// Validar Evento Generico
```

```
        public bool ValidarEventoGenerico(DateTime fecha, string turnoString, string
descripcion, string nombreCliente, string nombreServicio, int cantPersonasServicio, int
asistentes, string foto)
        {
            // Hace una validación básica de los datos ingresados para los eventos
            bool validado = false;
            Servicio servicio = BuscarServicio(nombreServicio);
            if (fecha > DateTime.Now &&
                turnoString != "" &&
                descripcion != "" &&
                nombreCliente != "" &&
                servicio != null &&
                cantPersonasServicio > 0 &&
                asistentes > 0 &&
                cantPersonasServicio <= asistentes)
            {
                validado = true;
            }
            return validado;
        }

        //Validar Evento Comun
        public bool ValidarEventoComun(DateTime fecha, string turnoString, string descripcion,
string nombreCliente, string nombreServicio, int cantPersonasServicio, int asistentes, string
foto, int duracion)
        {
            // Hace las validaciones específicas de los Eventos Comunes
            bool validado = false;
            if (ValidarEventoGenerico(fecha, turnoString, descripcion, nombreCliente,
nombreServicio, cantPersonasServicio, asistentes, foto) &&
                duracion > 0 &&
                duracion <= 4 &&
                asistentes <= 10 )
            {
                validado = true;
            }
            return validado;
        }

        // Validar Evento Premium
        public bool ValidarEventoPremium(DateTime fecha, string turnoString, string
descripcion, string nombreCliente, string nombreServicio, int cantPersonasServicio, int
asistentes, string foto)
        {
            // Hace la validación específica de los Eventos Premium
            bool validado = false;
            if (ValidarEventoGenerico(fecha, turnoString, descripcion, nombreCliente,
nombreServicio, cantPersonasServicio, asistentes, foto) &&
                asistentes <= 100)
            {
                validado = true;
            }
            return validado;
        }
```

```
public abstract class Evento: IComparable<Evento>
    {
        // Atributos
        protected DateTime fecha;
        protected Turno turno;
        protected string descripcion;
        protected string nombreCliente;
        protected List<ServicioContratado> serviciosContratados = new
List<ServicioContratado>();
        protected int asistentes;
        protected string foto;
        protected int id;
        protected static int idUlt = 1;
        protected decimal costoCalculado;

        // Propiedades
        public int ID
        {
            get { return id; }
        }
        public string Descripcion
        {
            get { return descripcion; }
        }
        public string NombreCliente
        {
            get { return nombreCliente; }
        }
        public string Foto
        {
            get {

                return "/img/" + this.foto; }
        }
        public DateTime Fecha
        {
            get { return fecha; }
        }
        public int Asistentes
        {
            get { return asistentes; }
        }
        public decimal CostoCalculado
        {
            get { return costoCalculado; }
        }
        public string DatosVarios
        {
            get { return this.id + " - " + this.descripcion + " - " +
this.fecha.ToShortDateString() + " - asistentes: " + this.asistentes ; }
        }

        // Constructor
        public Evento(DateTime fecha, Turno turno, string descripcion, string nombreCliente,
Servicio servicio, int cantPersonasServicio, int asistentes, string foto)
        {
            this.fecha = fecha;
            this.turno = turno;
            this.descripcion = descripcion;
            this.nombreCliente = nombreCliente;
            this.asistentes = asistentes;
```

```
        this.foto = foto;
        serviciosContratados.Add(new ServicioContratado(servicio, cantPersonasServicio));
        this.id = Evento.idUlt;
        Evento.idUlt++;
        this.costoCalculado = CalcularCosto();
    }

    // Agregar Servicio Contratado
    public bool AgregarServicioContratado(Servicio servicio, int cantPersonas)
    {
        bool agregado = false;
        int cantServicios = serviciosContratados.Count;
        ServicioContratado servicioNuevo = new ServicioContratado(servicio, cantPersonas);
        serviciosContratados.Add(servicioNuevo);
        if (cantServicios < serviciosContratados.Count)
        {
            ActualizarCosto(servicioNuevo.CalcularCosto());
            agregado = true;
        }
        return agregado;
    }

    // Calcular Costos de Servicios Contratados
    public decimal CalcularCostosServiciosContratados()
    {
        decimal costo = 0;
        foreach (ServicioContratado servicioContratado in serviciosContratados)
        {
            costo += servicioContratado.CalcularCosto();
        }
        return costo;
    }

    // Calcular Costo
    public abstract decimal CalcularCosto();

    // Actualizar Costo
    public abstract void ActualizarCosto(decimal costoServicio);

    // Datos de mis servicios
    public string DatosDeMisServicios()
    {
        string datos = "";
        foreach (ServicioContratado serv in serviciosContratados)
        {
            datos += serv.ToString()+ "\n";
        }
        return datos;
    }

    // Tengo Este Servicio
    public bool TengoEsteServicio(Servicio servicio)
    {
        bool loTengo = false;
        int i = 0;
        while (i < serviciosContratados.Count && loTengo == false)
        {
            if (servicio != null)
            {
                if (serviciosContratados[i].TengoEsteServicio(servicio))
                {
```

```
                    loTengo = true;

                }
                i++;
            }
        }
        return loTengo;
    }

    // Ordenar por fecha desc
    public int CompareTo(Evento other)
    {
        return this.fecha.CompareTo(other.Fecha) * -1;
    }

    internal List<ServicioContratado> ServiciosContratados()
    {
        return this.serviciosContratados;
    }
}
```

```
public class EventoComun:Evento
    {
        private int duracion;
        private static decimal montoFijoLimpieza = 100;

        // Propiedades
        public static decimal MontoFijoLimpieza
        {
            get { return montoFijoLimpieza; }
        }

        // Constructor
        public EventoComun(DateTime fecha, Turno turno, string descripcion, string
nombreCliente, Servicio servicio, int cantPersonasServicio, int asistentes, string foto, int
duracion) :base (fecha, turno, descripcion, nombreCliente, servicio, cantPersonasServicio,
asistentes, foto)
        {
            this.duracion = duracion;
        }

        // Cambiar Monto Fijo Limpieza
        public static bool CambiarMontoFijoLimpieza(decimal nuevoValor)
        {
            bool cambiado = false;
            decimal valorAnterior = EventoComun.montoFijoLimpieza;
            EventoComun.montoFijoLimpieza = nuevoValor;
            if (valorAnterior != EventoComun.montoFijoLimpieza)
            {
                cambiado = true;
            }
            return cambiado;
        }

        // Calcular Costo
        public override decimal CalcularCosto()
        {
            decimal costo = 0;
            costo += base.CalcularCostosServiciosContratados() + montoFijoLimpieza;
            return costo;
        }

        // Actualizar Costo
        public override void ActualizarCosto(decimal costoServicio)
        {
            costoCalculado += costoServicio;
        }

        // ToString
        public override string ToString()
        {
            return "COD: " + id + " Evento Común  " + " FECHA: " + fecha.ToShortDateString() +
" CLIENTE: " + nombreCliente + " DESC: " + descripcion + " PERSONAS: " + asistentes + " COSTO
TOTAL: (inc. limpieza) " + costoCalculado;
        }
```

```
public class EventoPremium:Evento
    {
        private static int porcentajeDeAumento = 15;

        // Propiedades
        public static int PorcentajeDeAumento
        {
            get { return porcentajeDeAumento; }
        }

        // Constructor
        public EventoPremium(DateTime fecha, Turno turno, string descripcion, string
nombreCliente, Servicio servicio, int cantPersonasServicio, int asistentes, string foto) :
base(fecha, turno, descripcion, nombreCliente, servicio, cantPersonasServicio, asistentes,
foto)
        {

        }

        // Cambiar Porcentaje De Aumento
        public static bool CambiarPorcentajeDeAumento(int nuevoValor)
        {
            bool cambiado = false;
            int valorAnterior = EventoPremium.porcentajeDeAumento;
            EventoPremium.porcentajeDeAumento = nuevoValor;
            if (valorAnterior != EventoPremium.porcentajeDeAumento)
            {
                cambiado = true;
            }
            return cambiado;
        }

        // Calcular Costo
        public override decimal CalcularCosto()
        {
            decimal costo = 0;
            costo += base.CalcularCostosServiciosContratados() * porcentajeDeAumento;
            return costo;
        }

        // Actualizar Costo
        public override void ActualizarCosto(decimal costoServicio)
        {
            costoCalculado += costoServicio * porcentajeDeAumento;
        }

        // ToString
        public override string ToString()
        {
            return "COD: " + id + " Evento Premium" + " FECHA: " + fecha.ToShortDateString() +
" CLIENTE: " + nombreCliente + " DESC: " + descripcion + " PERSONAS: " + asistentes + " COSTO
TOTAL: " + costoCalculado;
        }
```

```
public class Organizador : Usuario
    {
        private string nombre;
        private string telefono;
        private string direccion;
        private DateTime fechaRegistro;
        private List<Evento> eventos = new List<Evento>();

        public List<Evento> Eventos
        {
            get { return this.eventos; }
        }

        public string Nombre
        {
            get { return this.nombre; }
        }
        public string Telefono
        {
            get { return telefono; }
        }
        public string Direccion
        {
            get { return direccion; }
        }
        public DateTime FechaRegistro
        {
            get { return fechaRegistro; }
        }


        // Constructor
        public Organizador(string mail, string contrasenia, string nombre, string telefono,
string direccion) :base(mail, contrasenia)
        {
            this.nombre = nombre;
            this.telefono = telefono;
            this.direccion = direccion;
            fechaRegistro = DateTime.Today;
        }

        // Mi Rol
        public override string MiRol()
        {
            return "Organizador";
        }

        // Agregar Evento
        public bool AgregarEvento(Evento evento)
        {
            bool agregado = false;
            int cantEventos = eventos.Count;
            if (evento != null)
            {
                eventos.Add(evento);
            }
            if (cantEventos < eventos.Count)
            {
                agregado = true;
            }
            return agregado;
```

```
        }

        // Listar Eventos STRING
        public override string ListarEventos()
        {
            string lista = "---Mis eventos---" + "\n";
            decimal total = 0;
            foreach (Evento evento in eventos)
            {
                lista += evento.ToString() + "\n";
                total += evento.CostoCalculado;
            }
            lista += "TOTAL GENERADO: " + total;
            if (eventos.Count == 0)
            {
                lista = "---No tengo ningún evento---";
            }
            return lista;
        }

        // Total Generado por mis Eventos
        public decimal TotalGeneradoPorMisEventos()
        {
            decimal total = 0;
            foreach (Evento evento in eventos)
            {
                total += evento.CostoCalculado;
            }
            return total;
        }



        // Mis Datos
        public string MisDatos()
        {
            return "NOMBRE: " + nombre + " TEL: " + telefono + " DIR: " + direccion + "
REGISTRADO: " + fechaRegistro.ToShortDateString();
        }

        // To String
        public override string ToString()
        {
            return base.ToString() + " " + MisDatos();
        }


        public List<Evento> MisEventosEntreFechasOrdenados(DateTime fecha1, DateTime fecha2)
        {
            List<Evento> lista = new List<Evento>();
            foreach (Evento evento in eventos)
            {
                if (evento.Fecha >= fecha1 &&
                    evento.Fecha <= fecha2)
                {
                    lista.Add(evento);
                }
            }
            lista.Sort();
            return lista;
        }
```

```
public class Servicio
    {
        private string nombre;
        private string descripcion;
        private decimal precioPorPersona;

        // Propiedades
        public string Nombre
        {
            get { return nombre; }
        }
        public string Descripcion
        {
            get { return this.descripcion; }
        }
        public decimal PrecioPorPersona
        {
            get { return precioPorPersona; }
        }
        public string PrecioPorPersonaString
        {
            get { return precioPorPersona.ToString() + " por persona" ; }
        }

        // Constructor
        public Servicio(string nombre, string descripcion, decimal precioPorPersona)
        {
            this.nombre = nombre;
            this.descripcion = descripcion;
            this.precioPorPersona = precioPorPersona;
        }

        // ToString
        public override string ToString()
        {
            return nombre + " - $" + precioPorPersona + " por persona" + " - " + descripcion;
        }
```

```
public class ServicioContratado
    {
        private int cantPersonas;
        private Servicio servicio;
        private decimal precioPorPersona;

        public string NombreServicio
        {
            get { return this.servicio.Nombre; }

        }

        public decimal CostoPorPersona
        {
            get { return this.precioPorPersona; }

        }

        public int CantPersonas
        {
            get { return this.cantPersonas; }

        }



        // Constructor
        public ServicioContratado(Servicio servicio, int cantPersonas)
        {
            this.servicio = servicio;
            this.cantPersonas = cantPersonas;
            this.precioPorPersona = servicio.PrecioPorPersona;
        }


        // Calcular Costo
        public decimal CalcularCosto()
        {
            return cantPersonas * precioPorPersona;
        }

        // ToString
        public override string ToString()
        {
            return "SERVICIO: " + servicio.Nombre + " PERSONAS: " + cantPersonas + " PRECIO POR
PERSONA: " + precioPorPersona + " COSTO: " + CalcularCosto();
        }

        // Tengo Este Servicio
        public bool TengoEsteServicio(Servicio servicio)
        {
            bool loTengo = false;
            if (this.servicio == servicio)
            {
                loTengo = true;
            }
            return loTengo;
        }
```

```
public class Usuario: IComparable<Usuario>
    {
        private string mail;
        private string contrasenia;

        public string Mail
        {
            get { return mail; }
        }
        public string Contrasenia
        {
            get { return contrasenia; }
        }

        // Constructor
        public Usuario(string mail, string contrasenia)
        {
            this.mail = mail;
            this.contrasenia = contrasenia;
        }

        // Mi Rol
        public virtual string MiRol()
        {
            return "Administrador";
        }

        // Listar Eventos
        public virtual string ListarEventos()
        {
            return "--Soy Administrador, no tengo eventos--";
        }

        // To String
        public override string ToString()
        {
            return "MAIL: " + mail + " CONTRASEÑA: " + contrasenia;
        }

        // Validar Mail
        public static bool ValidarMail(string mail)
        {
            // Devuelve true si el mail cumple con los requerimientos especificados
            bool validado = false;
            if (mail != "")
            {
                bool tieneArroba = mail.Contains("@");
                bool arrobaNoAlPrincipio = mail[0] != '@';
                bool arrobaNoAlFinal = mail[mail.Length - 1] != '@';
                bool tienePunto = mail.Contains(".");
                bool puntoDespuesDeArroba = mail.IndexOf('.') > mail.IndexOf('@');
                bool dosCharDespuesDePunto = (mail.IndexOf('.') <= mail.Length - 3);
                if (tieneArroba && arrobaNoAlPrincipio && arrobaNoAlFinal && tienePunto &&
puntoDespuesDeArroba && dosCharDespuesDePunto)
                {
                    validado = true;
                }
            }
            return validado;
        }
```

```
// Validar Constraseña
public static bool ValidarContrasenia(string contrasenia)
{
    // Devuelve true si la contraseña cumple con los requerimientos especificados
    bool validado = false;
    if (contrasenia != "")
    {
        bool largo = contrasenia.Length >= 8;
        bool caracterEspecial = false;
        if (contrasenia.Contains("!") ||
            contrasenia.Contains(".") ||
            contrasenia.Contains(",") ||
            contrasenia.Contains(";"))
        {
            caracterEspecial = true;
        }
        bool mayuscula = false;
        if (contrasenia.ToLower() != contrasenia)
        {
            mayuscula = true;
        }

        if (largo && caracterEspecial && mayuscula)
        {
            validado = true;
        }
    }
    return validado;
}

// Validar Nombre
public static bool ValidarNombre(string nombre)
{
    // Devuelve true si el nombre cumple con los requerimientos especificados
    bool validado = false;
    if (nombre != "")
    {
        bool largo = nombre.Length >= 3;
        bool todasLetras = true;
        for (int i = 0; i < nombre.Length; i++)
        {
            if (!Char.IsLetter(nombre[i]))
            {
                todasLetras = false;
                break;
            }
        }
        if (largo && todasLetras)
        {
            validado = true;
        }
    }
    return validado;
}

// Ordenar por mail
public int CompareTo(Usuario other)
{
    return this.mail.CompareTo(other.Mail);
}
```