UNIVERSIDAD ORT
Uruguay

Universidad ORT Uruguay
Facultad de Ingeniería

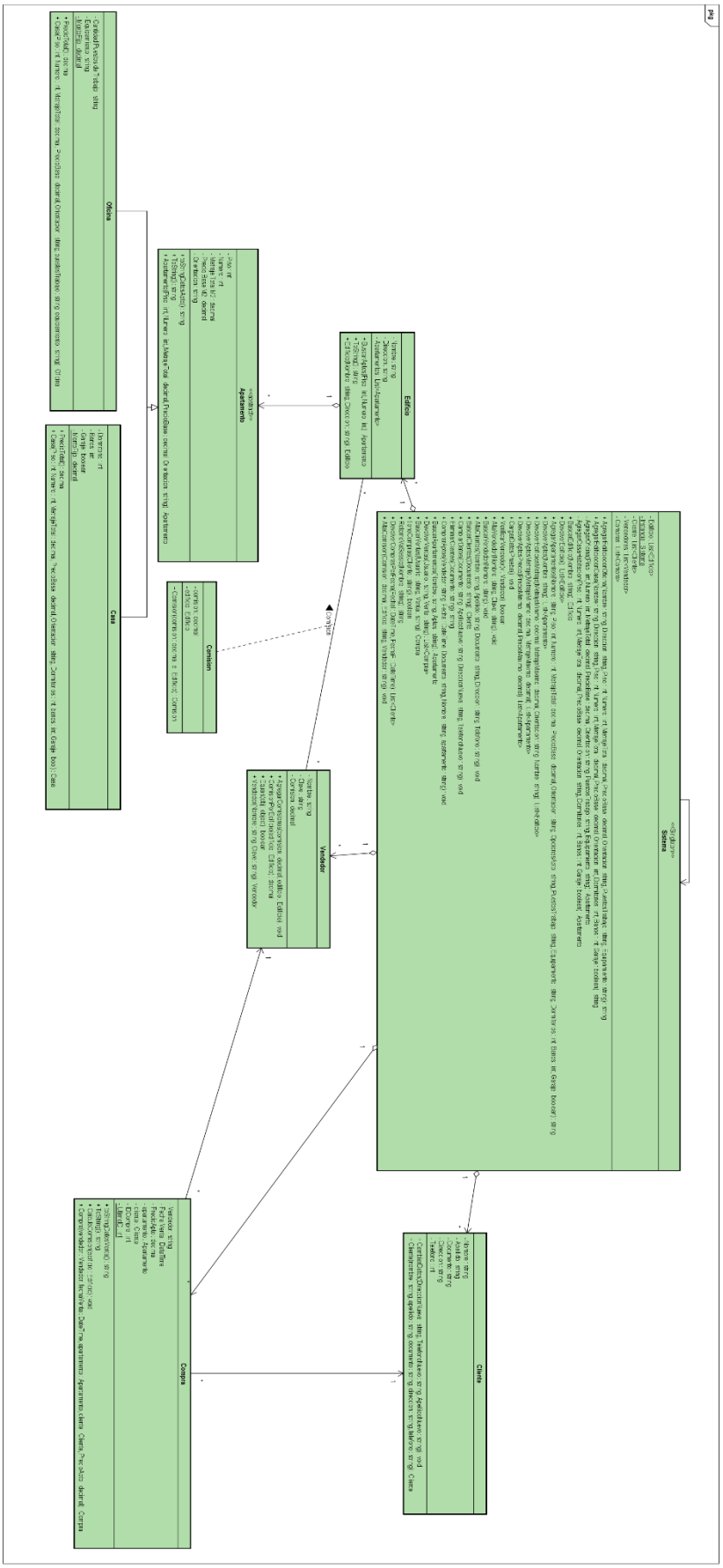
Obligatorio 2 Programación 2

Sebastian Alvez(232045)

Octubre 2018

Liliana Pino

Diseño UML



Código Fuente:

Edificio:

```
public class Edificio
{
    private string nombre;
    private string direccion;
    private List<Apartamento> ListaApartamentos;



    //Constructor
    public Edificio (string Nombre, string Direccion)
    {

        this.nombre = Nombre;
        this.direccion = Direccion;
        this.ListaApartamentos = new List<Apartamento>();

    }

    public List<Apartamento> ListaApartamentos1
    {
        get
        {
            return ListaApartamentos;
        }
    }

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


    }
    public string toStringDatos
    {
        get
        {
            return this.ToString();
        }
    }

    //Metodo que busca si el aparatemtno ingresado ya existe.
    public Apartamento BuscarAptos( int Piso, int Numero)
    {

        bool Existe = false;
        Apartamento a = null;
```

```
            int i = 0;

            while (i < ListaApartamentos1.Count && !Existe)
            {

                if (ListaApartamentos1[i].Piso == Piso && ListaApartamentos1[i].Numero == Numero)
                {

                    Existe = true;
                    a = ListaApartamentos1[i];

                }

                i++;
            }
            return a;

        }


        public override string ToString()
        {
            return this.nombre + " : " + this.direccion;
        }




    }
      }
```

Apartamento:

```
public abstract class Apartamento
    {
        private int piso;
        private int numero;
        private decimal metrajeTotal;
        private decimal precioBase;
        private string orientacion;


        //Constructor
        public Apartamento(int Piso, int Numero, decimal MetrajeTotal, decimal PrecioBase, string
Orientacion)
        {
            this.piso = Piso;
            this.numero = Numero;
            this.metrajeTotal = MetrajeTotal;
            this.precioBase = PrecioBase;
            this.orientacion = Orientacion;
        }

        public int Piso
        {
            get
```

```
        {
            return piso;
        }

        set
        {
            piso = value;
        }
    }

    public int Numero
    {
        get
        {
            return numero;
        }

        set
        {
            numero = value;
        }
    }

    public decimal MetrajeTotal
    {
        get
        {
            return metrajeTotal;
        }

        set
        {
            metrajeTotal = value;
        }
    }

    public decimal PrecioBase
    {
        get
        {
            return precioBase;
        }

        set
        {
            precioBase = value;
        }
    }

    public string Orientacion
    {
        get
        {
            return orientacion;
        }

        set
        {
            orientacion = value;
        }
    }
```

```
        public override string ToString()
        {
            return "Piso : " + this.piso + " , " + "Numero :"+this.numero + " , " + "Orientacion :" +
this.orientacion ;
        }


        //Devuelvo el ToString con lo que deseo mostrar.
        public string toStringDatosApto
        {
            get
            {
                return this.ToString();
            }
        }

        //Metodo para calcular el precio de los apartamentos,este es heredado a las clases hijas.
        public abstract decimal PrecioTotal();


    }
      }
```

Casa:

```
public class Casa : Apartamento
    {
        private int dormitorios;
        private int banos;
        private bool garaje;
        private int PrecioFijo = 200;


        //Constructor
        public Casa(int Piso, int Numero, decimal MetrajeTotal, decimal PrecioBase, string Orientacion,
int Dormitorios, int banos, bool Garaje) : base(Piso, Numero, MetrajeTotal, PrecioBase, Orientacion)
        {
            this.dormitorios = Dormitorios;
            this.banos = banos;
            this.garaje = Garaje;
        }

        public int Dormitorios
        {
            get
            {
                return dormitorios;
            }

            set
```

```
    {
        dormitorios = value;
    }
}

public int Banos
{
    get
    {
        return banos;
    }

    set
    {
        banos = value;
    }
}

public bool Garaje
{
    get
    {
        return garaje;
    }

    set
    {
        garaje = value;
    }
}


//Metodo que realiza el calculo para el precio del apartamento con destino a casa.
public override decimal PrecioTotal()
{
    decimal PrecioTotal = (PrecioBase * MetrajeTotal);

    switch (Dormitorios)
    {

        case 1:
            PrecioTotal = PrecioTotal +((PrecioTotal * 5) / 100);
            break;

        case 2:
            PrecioTotal = PrecioTotal +((PrecioTotal * 5) / 100);
            break;

        case 3:
            PrecioTotal = PrecioTotal + ((PrecioTotal * 10) / 100);
            break;

        case 4:
            PrecioTotal = PrecioTotal + ((PrecioTotal * 10) / 100);
            break;

        default:
            PrecioTotal = PrecioTotal + ((PrecioTotal * 20) / 100);

            break;
    }


    if (Garaje)
```

```
            {

                PrecioTotal = PrecioTotal + PrecioFijo;
            }
            if (Orientacion == "N" && Orientacion == "NE" && Orientacion == "NO")
            {

                PrecioTotal = PrecioTotal = (PrecioTotal * 15) / 100;
            }

            return PrecioTotal;
        }
    }
}
```

Oficina:

```
public class Oficina : Apartamento

    {
        private string puestosTrabajo;
        private string equipamiento;
        private int MontoFijo=100;


        //Construtor
        public Oficina(int Piso, int Numero, decimal MetrajeTotal, decimal PrecioBase, string
Orientacion, string puestosTrabajo, string equipamiento) : base(Piso, Numero, MetrajeTotal, PrecioBase,
Orientacion)
        {
            this.puestosTrabajo = puestosTrabajo;
            this.equipamiento = equipamiento;
        }



        public string Equipamiento
        {
            get
            {
                return equipamiento;
            }

            set
            {
                equipamiento = value;
            }
        }

        public string PuestosTrabajo
        {
            get
            {
                return puestosTrabajo;
            }

            set
```

```
            {
                puestosTrabajo = value;
            }
        }

        //Metodo que calcula el precio de las ordicinas
        public override  decimal PrecioTotal()
        {
            decimal PrecioTotal = (PrecioBase * MetrajeTotal)+ ( Convert.ToInt32(PuestosTrabajo)*
MontoFijo) ;
            if (Equipamiento == "Si") {

                PrecioTotal = (PrecioTotal * 10) / 100;
            }
            return PrecioTotal;
        }
    }
}
```

Cliente:

```
public  class Comision
    {
        decimal Pocentajecomision;
        private  Edificio edificio;

        public Comision(decimal comision, Edificio e)
        {
            this.Pocentajecomision = comision;
            this.edificio = e;
        }

        public decimal Pocentajecomision1
        {
            get
            {
                return Pocentajecomision;
            }

            set
            {
                Pocentajecomision = value;
            }
        }

        public Edificio Edificio
        {
            get
            {
                return edificio;
            }

            set
            {
                edificio = value;
            }
        }
```

```
    }
}
```

Comisión:

```
public  class Comision
    {
        decimal Pocentajecomision;
        private  Edificio edificio;

        public Comision(decimal comision, Edificio e)
        {
            this.Pocentajecomision = comision;
            this.edificio = e;
        }

        public decimal Pocentajecomision1
        {
            get
            {
                return Pocentajecomision;
            }

            set
            {
                Pocentajecomision = value;
            }
        }

        public Edificio Edificio
        {
            get
            {
                return edificio;
            }

            set
            {
                edificio = value;
```

```
            }
        }
    }
}
```

Compra:

```
public class Compra {


        private Vendedor vendedor;
        private Cliente cliente;
        private DateTime FechaVenta;
        private decimal precioApto;
        private Apartamento apartamento;
        private int IDCompra;
        private static int ultimoID;
        private decimal ComisionFinal;

        public Vendedor Vendedor
        {
            get
            {
                return vendedor;
            }

            set
            {
                vendedor = value;
            }
        }

        public DateTime FechaVenta1
        {
            get
```

```
            {
                return FechaVenta;
            }

            set
            {
                FechaVenta = value;
            }
        }

        public decimal PrecioApto
        {
            get
            {
                return precioApto;
            }

            set
            {
                precioApto = value;
            }
        }

        public Apartamento Apartamento
        {
            get
            {
                return apartamento;
            }

            set
            {
                apartamento = value;
            }
        }

        public Cliente Cliente
        {
            get
            {
                return cliente;
            }

            set
            {
                cliente = value;
            }
        }

        public Compra(Vendedor vendedor, DateTime fechaVenta, Apartamento apartamento,Cliente
cliente,decimal PrecioApto)
        {
            this.Vendedor = vendedor;
            FechaVenta1 = fechaVenta;
            this.precioApto = PrecioApto;
            this.Apartamento = apartamento;
            this.Cliente = cliente;
            this.UltimoID = UltimoID + 1;
            this.IDCompra = UltimoID;
            this.ComisionFinal = vendedor.PorcentajeComision1;


        }
```

```
public override string ToString()
{

    return "venta  " + this.IDCompra;

}


//Devuelvo el ToString con lo que deseo mostrar.
public string toStringDatosVenta
{
    get
    {
        return this.ToString();
    }
}

public int IDCompra1
{
    get
    {
        return IDCompra;
    }

    set
    {
        IDCompra = value;
    }
}

public int UltimoID
{
    get
    {
        return ultimoID;
    }

    set
    {
        ultimoID = value;
    }
}



public decimal ComisionFinal1
{
    get
    {
        return ComisionFinal;
    }

    set
    {
        ComisionFinal = value;
    }
}

public void CalculoComision(Edificio edificio) {


    decimal PorcentajeComision = vendedor.ComisionPorEdificio(edificio);
```

```
            ComisionFinal = ((Apartamento.PrecioTotal() * PorcentajeComision) / 100);

        }
    }
}
```

Vendedor:

```
    public class Vendedor
    {
        private string nombre;
        private string clave;
        private decimal PorcentajeComision;
        private List<Comision> Comision;


        public Vendedor(string Nombre,string  Clave) {

            this.nombre = Nombre;
            this.clave = Clave;
            this.PorcentajeComision = PorcentajeComision1;
```

```
}

public Vendedor()
{
}

public string Nombre
{
    get
    {
        return nombre;
    }

    set
    {
        nombre = value;
    }
}

public string Clave
{
    get
    {
        return clave;
    }

    set
    {
        clave = value;
    }
}

public List<Comision> Comision1
{
    get
    {
        return Comision;
    }

    set
    {
        Comision = value;
    }
}

public decimal PorcentajeComision1
{
    get
    {
        return PorcentajeComision;
    }

    set
    {
        PorcentajeComision = value;
    }
}

public void AgregarComisiones(decimal comision, Edificio edificio) {

    Comision c = new Comision(comision, edificio);

    if (Comision1 == null)
```

```
            {
                Comision1 = new List<Comision>();
                Comision1.Add(c);
            }
            else {
                Comision1.Add(c);
            }

        }

        public decimal ComisionPorEdificio(Edificio edificio) {


            int i = 0;
            decimal comisionEdificio = 0;
            bool Bandera = false;

            while (i< Comision1.Count  && !Bandera) {

                if (Comision1[i].Edificio.Nombre == edificio.Nombre) {

                    comisionEdificio = Comision1[i].Pocentajecomision1;
                    Bandera = true;


                }
                i++;


            }
            return comisionEdificio;

        }

        public override bool Equals(object obj)
        {
            bool esIgual = false;
            if (obj != null)
            {
                if (obj is Vendedor)
                {
                    esIgual = ((Vendedor)obj).nombre == this.Nombre  &&((Vendedor)obj).clave ==
this.Clave;
                }
            }
            return esIgual;
        }


    }
}
```

Sistema:

```
public class Sistema
    {
        private List<Edificio> Edificios = new List<Edificio>();
        private List<Vendedor> Vendedores { get; set; } = new List<Vendedor>();
        public List<Cliente> Clientes = new List<Cliente>();
        public List<Compra> Compras = new List<Compra>();
```

```
        private static Sistema instancia;

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

        private Sistema()
        {
            CargarDatosPrueba();
        }


        //Region Edificios


        #region
        //Agrega un edificio con un primer aparatamento.
        public string AgregarEdificioconOficina(string Nombre, string Direccion, int Piso, int Numero,
decimal MetrajeTotal, decimal PrecioBase, string Orientacion, string PuestosTrabajo, string
Equipamiento)
        {
            string Mensaje = "";
            if (Nombre != "" && Direccion != "")
            {
                if (this.BuscarEdificio(Nombre) == null)
                {
                    Edificio e = new Edificio(Nombre, Direccion);
                    Edificios.Add(e);
                    e.ListaApartamentos1.Add(AgregarOficina(Piso, Numero, MetrajeTotal, PrecioBase,
Orientacion, PuestosTrabajo, Equipamiento));

                    Mensaje = "Se agrego Correctamente el Edificio.";
                }
                else
                {

                    Mensaje = "El edificio ingresado ya existe.";
                }
            }
            else
            {

                Mensaje = "Hay campos vacios,Verifique!";

            }

            return Mensaje;
        }

        public string AgregarEdificioconCasa(string Nombre, string Direccion, int Piso, int Numero,
decimal MetrajeTotal, decimal PrecioBase, string Orientacion, int Dormitorios, int Banos, bool Garaje)
        {
            string Mensaje = "";
            if (Nombre != "" && Direccion != "")
            {
```

```
                if (this.BuscarEdificio(Nombre) == null)
                {
                    Edificio e = new Edificio(Nombre, Direccion);
                    Edificios.Add(e);

                    e.ListaApartamentos1.Add(AgregarCasaHabitacion(Piso, Numero, MetrajeTotal,
PrecioBase, Orientacion, Dormitorios, Banos, Garaje));

                    Mensaje = "Se agrego Correctamente el Edificio.";
                }
                else
                {

                    Mensaje = "El edificio ingresado ya existe.";
                }
            }
            else
            {

                Mensaje = "Hay campos vacios,Verifique!";

            }

            return Mensaje;
        }

        //En estos metodos creo un Apartamentos con el tipo  Casa o una Oficina.
        private Apartamento AgregarOficina(int Piso, int Numero, decimal MetrajeTotal, decimal
PrecioBase, string Orientacion, string PuestosTrabajo, string Equipamiento)
        {
            Apartamento a = new Oficina(Piso, Numero, MetrajeTotal, PrecioBase, Orientacion,
PuestosTrabajo, Equipamiento);

            return a;
        }

        private Apartamento AgregarCasaHabitacion(int Piso, int Numero, decimal MetrajeTotal, decimal
PrecioBase, string Orientacion, int Dormitorios, int Banos, bool Garaje)
        {
            Apartamento a = new Casa(Piso, Numero, MetrajeTotal, PrecioBase, Orientacion, Dormitorios,
Banos, Garaje);

            return a;
        }


        //Metodo que busca si el edificio ingresado  ya existe.
        public Edificio BuscarEdificio(string Nombre)
        {

            bool Existe = false;
            Edificio e = null;
            int i = 0;

            while (i < Edificios.Count && !Existe)
            {

                if (Edificios[i].Nombre.ToUpper() == Nombre.ToUpper())
                {

                    Existe = true;
                    e = Edificios[i];

                }
```

```
                i++;
            }
            return e;

        }



        //Retorna la lista de Edificios
        public List<Edificio> DevolverEdificios
        {
            get
            {
                return Edificios;
            }

        }
        #endregion



        //Region Aparatamentos
        #region

        //En este metodos creo un Apartamentos con el tipo  Casa o una Oficina,contando con ciertas
validaciones.
        public string AgregarApartamentos(string Nombre, int Piso, int Numero, decimal MetrajeTotal,
decimal PrecioBase, string Orientacion, string OpcionesApto, string PuestosTrabajo, string
Equipamiento, int Dormitorios, int Banos, bool Garaje)
        {
            string Mensaje = "";
            Edificio edi = BuscarEdificio(Nombre);
            List<Apartamento> Retorno = null;

            if (edi != null)
            {
                if (edi.BuscarAptos(Piso, Numero) == null)
                {
                    if (OpcionesApto == "Oficina")
                    {
                        if (Piso != 0 && Numero != 0 && MetrajeTotal != 0 && PrecioBase != 0 &&
PuestosTrabajo != "")
                        {
                            Apartamento a = new Oficina(Piso, Numero, MetrajeTotal, PrecioBase,
Orientacion, PuestosTrabajo, Equipamiento);
                            edi.ListaApartamentos1.Add(a);
                            Retorno = edi.ListaApartamentos1;
                            Mensaje = "Se agrego un apartamento con destino a oficina con exito";
                        }
                        else
                        {
                            Mensaje = "Se ingresaron campos vacios o nulos,VERIFIQUE!";
                        }
                    }

                    if (OpcionesApto == "CasaHabitacion")
                    {
                        if (Piso != 0 && Numero != 0 && MetrajeTotal != 0 && PrecioBase != 0)
                        {
                            Apartamento a = new Casa(Piso, Numero, MetrajeTotal, PrecioBase,
Orientacion, Dormitorios, Banos, Garaje);
                            edi.ListaApartamentos1.Add(a);
```

```
                        Retorno = edi.ListaApartamentos1;
                        Mensaje = "Se agrego un apartamento con destino a casa con exito";
                    }
                    else
                    {
                        Mensaje = "Se ingresaron campos vacios o nulos,VERIFIQUE!";
                    }
                }
            }
            else
            {
                Mensaje = "No se agrego el apartamento,ya que existe uno con el mismo piso y
numero.";

            }

        }


        return Mensaje;
    }


    //Deuelvo una lista de Aparatamentos.

    public List<Apartamento> DevolverAptos(string Nombre)
    {

        Edificio edi = BuscarEdificio(Nombre);
        List<Apartamento> Retorno = null;

        if (edi != null)
        {

            Retorno = edi.ListaApartamentos1;
        }

        return Retorno;
    }


    #endregion


    //Region Filtrar edificios por metraje.
    #region

    //Metodo que vuelve una lista de edificios que cumplan con los requisitos del metraje.
    public List<Edificio> DevolverEdificiosMetraje(decimal MetrajeMinimo, decimal MetrajeMaximo,
string Orientacion, string Nombre)
    {
        List<Edificio> FiltroMetraje = new List<Edificio>();
        bool esta = false;
        if (MetrajeMinimo != 0 && MetrajeMaximo != 0)
        {

            foreach (Edificio e in Edificios)
            {

                for (int i = 0; i < e.ListaApartamentos1.Count; i++)
                {
                    if (e.ListaApartamentos1[i].MetrajeTotal >= MetrajeMinimo &&
e.ListaApartamentos1[i].MetrajeTotal <= MetrajeMaximo
                        && e.ListaApartamentos1[i].Orientacion == Orientacion && !esta)
```

```
                    {
                        {
                            FiltroMetraje.Add(e);
                            esta = true;
                        }


                    }

                }
            }

        }

        return FiltroMetraje;
    }


    #endregion


    //Region Filtrar Apartamentos por metraje.
    #region

    //Metodo que vuelve una lista de apartamentos de todos los edificios que cumplan con los
requisitos del metraje.
    public List<Apartamento> DevolverAptosMetraje(decimal MetrajeMinimo, decimal MetrajeMaximo)
    {
        List<Apartamento> FiltroApartamentosMetraje = new List<Apartamento>();
        if (MetrajeMinimo != 0 && MetrajeMaximo != 0)
        {

            foreach (Edificio e in Edificios)
            {

                for (int i = 0; i < e.ListaApartamentos1.Count; i++)
                {
                    if (e.ListaApartamentos1[i].MetrajeTotal >= MetrajeMinimo &&
e.ListaApartamentos1[i].MetrajeTotal <= MetrajeMaximo)
                    {

                        FiltroApartamentosMetraje.Add(e.ListaApartamentos1[i]);

                    }
                }
            }
        }
        return FiltroApartamentosMetraje;
    }


    #endregion

//Region Filtrar Apartamentos por precio.
    #region

    //Metodo que vuelve una lista de apartamentos de todos los edificios que cumplan con los
requisitos del precio.
    public List<Apartamento> DevolverAptosPrecio(decimal PrecioMinimo, decimal PrecioMaximo)
    {
        Apartamento a = null;
        List<Apartamento> FiltroApartamentosPrecio = new List<Apartamento>();

        if (PrecioMinimo != 0 && PrecioMaximo != 0)
```

```
            {

                foreach (Edificio e in Edificios)
                {

                    for (int i = 0; i < e.ListaApartamentos1.Count; i++)
                    {
                        if (e.ListaApartamentos1[i].PrecioTotal() >= PrecioMinimo &&
e.ListaApartamentos1[i].PrecioTotal() <= PrecioMaximo)
                        {


                            FiltroApartamentosPrecio.Add(e.ListaApartamentos1[i]);

                            a = FiltroApartamentosPrecio[i];


                        }
                    }
                }
            }
            return FiltroApartamentosPrecio;
        }
        #endregion

        public void CargarDatosPrueba()
        {
            this.AltaVendedor("vend1", "vend1111");
            this.AltaVendedor("vend2", "vend2222");
            this.AltaClientes("Juan", "Perez", "12345678", "Mexico 1668", "091234567");
            this.AltaClientes("Maria", "Lopez", "23456789", "Grecia 1223", "092345678");
            this.AgregarEdificioconCasa("Hilton", "Mexico 1668", 1, 1, 5000, 4000, "S", 3, 2, true);
            this.AgregarEdificioconOficina("Radisson", "Grecia 1223", 1, 1, 5500, 3200, "N", "350",
"Si");
            this.AgregarApartamentos("Hilton", 1, 2, 5000, 4000, "SE", "CasaHabitacion", "0", "No", 3,
1, false);
            this.AgregarApartamentos("Hilton", 2, 2, 3000, 2500, "NE", "CasaHabitacion", "0", "No", 1,
1, true);
            this.AgregarApartamentos("Hilton", 2, 3, 6000, 5000, "O", "CasaHabitacion", "0", "No", 3,
2, true);
            this.AgregarApartamentos("Hilton", 4, 2, 4500, 4000, "E", "CasaHabitacion", "0", "No", 2,
1, true);
            this.AgregarApartamentos("Hilton", 3, 1, 4000, 3500, "NO", "CasaHabitacion", "0", "No", 2,
2, false);
            this.AgregarApartamentos("Hilton", 2, 5, 4000, 3500, "SE", "Oficina", "200", "No", 0, 0,
false);
            this.AgregarApartamentos("Hilton", 3, 7, 5500, 5000, "NE", "Oficina", "350", "Si", 0, 0,
false);
            this.AgregarApartamentos("Hilton", 4, 1, 4500, 4000, "SO", "Oficina", "250", "Si", 0, 0,
false);

            this.AgregarApartamentos("Radisson", 1, 2, 5000, 4000, "SE", "CasaHabitacion", "0", "No",
3, 1, false);
            this.AgregarApartamentos("Radisson", 2, 2, 3000, 2500, "NE", "CasaHabitacion", "0", "No",
1, 1, true);
            this.AgregarApartamentos("Radisson", 2, 3, 6000, 5000, "O", "CasaHabitacion", "0", "No", 3,
2, true);
            this.AgregarApartamentos("Radisson", 4, 2, 4500, 4000, "E", "CasaHabitacion", "0", "No", 2,
1, true);
            this.AgregarApartamentos("Radisson", 3, 1, 4000, 3500, "NO", "CasaHabitacion", "0", "No",
2, 2, false);
            this.AgregarApartamentos("Radisson", 2, 5, 4000, 3500, "SE", "Oficina", "200", "No", 0, 0,
false);
```

```
            this.AgregarApartamentos("Radisson", 3, 7, 5500, 5000, "NE", "Oficina", "350", "Si", 0, 0,
false);
            this.AgregarApartamentos("Radisson", 4, 1, 4500, 4000, "SO", "Oficina", "250", "Si", 0, 0,
false);

            this.ComprarAptos("vend1", DateTime.Now, "12345678", "Radisson", "Piso : " + 1 + " , " +
"Numero :" + 2 + " , " + "Orientacion :" + "SE");
            this.ComprarAptos("vend1", DateTime.Now, "12345678", "Radisson", "Piso : " + 4 + " , " +
"Numero :" + 1 + " , " + "Orientacion :" + "SO");
            this.ComprarAptos("vend1", DateTime.Now, "23456789", "Hilton", "Piso : " + 4 + " , " +
"Numero :" + 1 + " , " + "Orientacion :" + "SO");
            this.ComprarAptos("vend2", DateTime.Now, "23456789", "Hilton", "Piso : " + 1 + " , " +
"Numero :" + 2 + " , " + "Orientacion :" + "SE");
            this.ComprarAptos("vend2", DateTime.Now, "23456789", "Hilton", "Piso : " + 3 + " , " +
"Numero :" + 7 + " , " + "Orientacion :" + "NE");

            this.AltaComision(5, "Radisson", "vend1");
            this.AltaComision(2, "Hilton", "vend1");
        }

        public bool VerificarVendedor(Vendedor v)
        {
            bool usuarioCorrecto = false;
            if (this.Vendedores.Contains(v))
            {
                usuarioCorrecto = true;
            }
            return usuarioCorrecto;
        }

        public void AltaVendedor(string Nombre, string Clave)
        {
            Vendedor v = BuscarVendedor(Nombre);
            if (v == null)
            {
                v = new Vendedor(Nombre, Clave);
                this.Vendedores.Add(v);
            }
        }

        public Vendedor BuscarVendedor(string Nombre)
        {
            Vendedor v = null;
            int i = 0;
            while (i < this.Vendedores.Count && v == null)
            {
                if (Vendedores[i].Nombre == Nombre)
                {
                    v = Vendedores[i];

                }
                i++;
            }
            return v;
        }

public void AltaClientes(string Nombre, string Apellido, string Documento, string Direccion, string
Telefono)
        {
            Cliente c = BuscarClientes(Documento);
            if (c == null)
            {
                c = new Cliente(Nombre, Apellido, Documento, Direccion, Telefono);
                this.Clientes.Add(c);
```

```
        }
    }
```

```
    public Cliente BuscarClientes(string Documento)
    {
        Cliente c = null;
        int i = 0;
        while (i < this.Clientes.Count && c == null)
        {
            if (Clientes[i].Documento == Documento)
            {
                c = Clientes[i];

            }
            i++;
        }
        return c;
    }
    public void CambiarDatos(string Documento, string ApellidoNuevo, string DireccionNueva, string
TelefonoNuevo)
    {

        Cliente c = BuscarClientes(Documento);

        if (c != null)
        {

            c.CambiarDatos(ApellidoNuevo, DireccionNueva, TelefonoNuevo);

        }
    }


    public string EliminarClientes(string Documento)
    {

        Cliente c = BuscarClientes(Documento);
        string Mensaje = "";

        if (c != null)
        {

            if (tieneCompras(Documento) == false)
            {
                Clientes.Remove(c);
                Mensaje = "El cliente se removio con exito";
            }
            else
            {
                Mensaje = "No se puede eliminar el cliente,porque tiene compras realizadas!";
            }
```

```
            }
            return Mensaje;
        }

        public void ComprarAptos(string Vendedor, DateTime Fecha, string Documento, string Nombre,
string apartamento)
        {

            Cliente c = BuscarClientes(Documento);
            Edificio e = BuscarEdificio(Nombre);
            Vendedor v = BuscarVendedor(Vendedor);
            Apartamento a = BuscarApartamentos(Nombre, apartamento);

            if (c != null && e != null)
            {

                Compra co = new Compra(v, Fecha, a, c,a.PrecioBase);
                Compras.Add(co);

                e.ListaApartamentos1.Remove(a);
            }

        }

        public Apartamento BuscarApartamentos(string Nombre, string Aptos)
        {

            bool Existe = false;
            Edificio e = BuscarEdificio(Nombre);
            Apartamento a = null;
            int i = 0;

            while (i < e.ListaApartamentos1.Count && !Existe)
            {

                if (e.ListaApartamentos1[i].toStringDatosApto.Trim() == Aptos.Trim())
                {

                    Existe = true;

                    a = e.ListaApartamentos1[i];
```

```
        }

        i++;
    }
    return a;

}
```

```
public List<Compra> DevolverVentas(string Usuario, string Venta)
{
    Compra com = BuscarVentas(Usuario, Venta);
    Comision c = null;

    List<Compra> Retorno = new List<Compra>();

    if (com != null)
    {

        Retorno.Add(com);



    }

    return Retorno;
}
```

```
public Compra BuscarVentas(string Usuario, string Venta)
{

    bool Existe = false;
    Compra com = null;
    int i = 0;

    while (i < Compras.Count && !Existe)
    {

        if (Compras[i].Vendedor.Nombre == Usuario)
        {
            if (Compras[i].toStringDatosVenta == Venta)
            {

                Existe = true;
                com = Compras[i];
            }
```

```
            }
            i++;
        }
        return com;

    }

    public bool tieneCompras(string Cliente)
    {

        bool tiene = false;
        int i = 0;
        while (i < Compras.Count && !tiene)
        {
            if (Compras[i].Cliente.Documento == Cliente)
            {

                tiene = true;

            }
            i++;
        }

        return tiene;

    }


    public string RetornoVaSessio(string Nombre)
    {

        string VariableSession = Nombre;

        return VariableSession;

    }



    public List<Cliente> DevolverComprasPorFecha(DateTime FechaI, DateTime FechaF)
    {

        Cliente c = null;
        List<Cliente> RetornoClientes = new List<Cliente>();


        int i = 0;

        while (i < Compras.Count)
        {

            if (Compras[i].FechaVenta1 >= FechaI && Compras[i].FechaVenta1 <= FechaF)
            {
                if (!RetornoClientes.Contains(Compras[i].Cliente))
                {
                    c = Compras[i].Cliente;
                    RetornoClientes.Add(c);
                   RetornoClientes = RetornoClientes.OrderBy(o => o.Nombre).ToList();
}
}

            i++;

        }
```

```
            return RetornoClientes;
        }



        public void AltaComision(decimal Comision, string Edificio, string Vendedor)
        {
            int i = 0;
            bool bandera = false;
            Edificio e = BuscarEdificio(Edificio);


            while (i < Compras.Count && !bandera) {

                if (Compras[i].Vendedor.Nombre == Vendedor) {

                    Vendedores[i].AgregarComisiones(Comision, e);
                    Compras[i].CalculoComision(e);
                    bandera = true;


                }
                i++;

            }


        }
    }
    }
```

Casos de prueba:

Escenario de Test	Datos utilizados	Resultado Esperado	Resultado obtenido	Estado (F=Falla/P=Pasa)
Ingreso Usuario correcto	Vend1(vend1111 (contraseña)) Vend2(vend2222 (contraseña))	Ingresa correctamente	Ingreso correctamente	P
Ingreso usuario incorrecto (Contraseña o usuario)	Vend12(vend1111(contraseña)) Vend21(vend2252(contraseña))	Muestra mensaje de que los usuarios no son correctos	Muestra el mensaje correcto	P
Ingreso datos vacíos		Muestra mensaje de que los datos están vacíos	Muestra el mensaje correcto	P
El usuario logueado cierra sesión.		El usuario cierra sesión correctamente	Función Correcta	P
Se quiere cerrar sesión y no hay usuarios logueados		Muestra un mensaje de que no hay usuarios logueados.	Muestra el mensaje correcto	P

Ingreso a dar de alta, modificar o eliminar un cliente, mientras hay un vendedor logueado.		Se redireccióna para cerrar sesión.	Realiza la esperado.	P
Se da de alta un usuario		Muestra mensaje de que se dio de alta un usuario correctamente	Muestra mensaje correcto	P
Se intenta dar de alta un usuario existente		Muestra mensaje de que el usuario ya existe.	Muestra mensaje correcto	P
Se intenta dar de alta un usuario con datos vacíos		Muestra mensaje de que los datos están vacíos	Muestra mensaje correcto	P
Se cambia los datos de la persona, excepto el documento.		Muestra mensaje de que los datos se cambiaron con éxito	Muestra mensaje correcto	P
Se intenta cambiar datos a vacíos		Muestra mensaje de que los datos están vacíos	Muestra mensaje correcto	P
Se elimina un cliente		Muestra mensaje de que el cliente fue eliminado.	Muestra mensaje correcto	P
Se intenta eliminar un cliente con ventas		Muestra mensaje de que el cliente no se puede eliminar porque tiene ventas.	Muestra mensaje correcto	P
Venta de un Apartamento		Muestra un mensaje de que se vendió un apartamento con éxito y automáticamente el apartamento seleccionado ya no estará disponible para otra venta.	Muestra mensaje correcto y la función se realiza con éxito.	P
Venta de un apartamento sin un vendedor logueado		Se redireccióna para iniciar sesión.	Se redireccióna para iniciar sesión.	P
Venta de un apartamento con datos vacíos		Muestra mensaje de que los datos están vacíos	Muestra mensaje correcto	P
Ver información de las ventas del vendedor logueado		Muestra toda la información de las ventas(Cliente, Apartamento y comisión)	Muestra la información correcta	P
Ver información de las ventas sin ningún vendedor logueado		Se redireccióna para iniciar sesión.	Se redireccióna para iniciar sesión.	P

Ver los clientes que compraron Apartamentos en un rango determinado de fechas {		Se muestra los datos de los clientes que realizaron las compras de apartamentos entre las fechas dadas	Muestra los datos correcto	P
Se ingresan fechas vacías		Muestra mensaje de que los datos están vacíos	Muestra mensaje correcto	P
La primera fecha ingresada es mayor que la segunda }		Muestra un mensaje de que la fecha inicial es mayor a la final, que verifique.	Muestra mensaje correcto	P