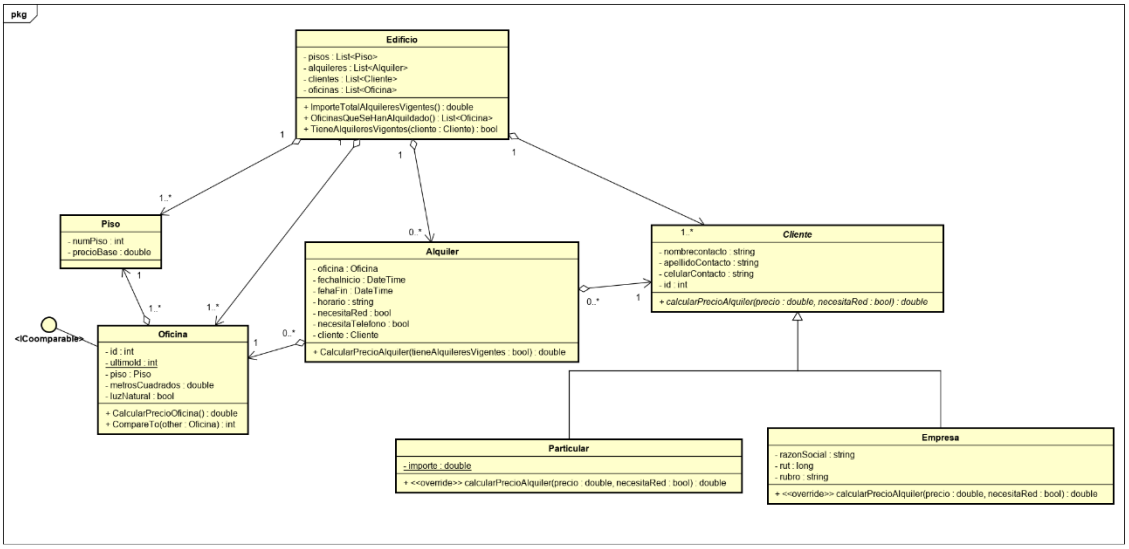
Solución Examen Octubre 2021



Devolver el importe total de los alquileres de oficinas que están vigentes

Edificio

```
  public double ImporteTotalAlquileresVigentes()
        {
            double importeTotal = 0;
            foreach (Alquiler miAlquiler in alquileres)
            {
                if(miAlquiler.FechaFin > DateTime.Now){
                    bool tieneAlquileresVigentes =
this.ObtenerAlquileresVigentes(miAlquiler.Cliente);
                    importeTotal +=
miAlquiler.CalcularPrecioAlquiler(tieneAlquileresVigentes);
                }
            }
            return importeTotal;
        }


public bool TieneAlquileresVigentes(Cliente cliente)
        {
            bool tieneVigentes = false;
            int i = 0;
            while(i < alquileres.count && !tieneVigentes){
                if(alquileres[i].Cliente.Equals(cliente) &&
alquileres[i].FechaFin > DateTime.Now){
                    tieneVigentes = true;
                }
                I++;
            }
            return tieneVigentes;
        }
```

Alquiler

```
   public double CalcularPrecioAlquiler(bool tieneAlquileresVigentes)
```

```
        {
            double precio = this.oficina.CalcularPrecioOficina();
            if(this.necesitaTelefono){
                precio = precio * 1.10;
            }
            if(tieneAlquileresVigentes){
                precio = precio * 0.95;
            }
            precio = this.cliente.CalcularPrecioFinal(precio, necesitaRed);
            return precio;
        }
```

Oficina

```
    public double CalcularPrecioOficina()
        {
         return  this.piso.PrecioBase * this.metrosCuadrados;
        }
```

Cliente

public abstract double CalcularPrecioAlquiler(double precio,bool necesitaRed);

Particular

```
public override double CalcularPrecioAlquiler(double precio, bool necesitaRed)
        {
            double precioFinal = precio;
            if (necesitaRed)
            {
                precioFinal += Particular.importe;
            }
            return precioFinal;
        }
```

Empresa

```
  public override double CalcularPrecioAlquiler(double precio, bool necesitaRed)
        {
            double precioFinal = precio;
            if(necesitaRed){
                precioFinal = precioFinal * 1.15;
            }
            if(rubro.trim().ToUpper() =="TECNOLOGIA"){
                precioFinal = precioFinal * 0.9*;
            }
            return precioFinal;
        }
```

2-Devolver las oficinas que se han alquilado por los menos una vez, ordenados por piso y metros cuadrados en forma descendente La oficina no puede repetirse, es decir si una oficina se alquiló más de una vez se debe retornar una sola vez. (25 puntos)

Edificio

```
  public List<Oficina>OficinasQueSeHanAlquilado(){
```

```
            List<Oficina> oficinasAlquiladas = new List<Oficina>();
            foreach(Alquiler miAlquiler in this.Alquileres){
                if(oficinasAlquiladas.Contains(miAlquiler.Oficina)){
                    oficinasAlquiladas.Add(miAlquiler.oficina);
                }
            }
            oficinasAlquiladas.Sort();
            return oficinasAlquiladas;
        }
```

Oficina

public class Oficina : IComparable<Oficina>{

```
public int CompareTo(Oficina other)
        {
            int comparación = (this.piso – other.Piso)*-1;
            if(comparacion == 0){
                comparacion = (this.metrosCuadrados – other.MetrosCuadrados)*-1;
            }
            return comparación;

}
```

MVC

Dada un identificador de un cliente mostrar todos los alquileres que ha realizado. Se puede asumir para la solución de este requerimiento que existe en el dominio un método llamado AlquileresCliente(int idCliente) el cual retorna una lista de alquileres.(15 puntos)

```
@{
    ViewBag.Title = "AlquileresCliente";
}


<h2>Alquileres cliente</h2>

<form action="~/Cliente/ObtenerAlquileresCliente" method="post">
    <select name="idCliente">
        <option value="-1"">Seleccione un cliente</option>

        @foreach (Cliente cliente in Sistema.Instancia.Clientes)
        {
            <option value="@cliente.Id"> @cliente.NombreContacto</option>
        }
        </select>
        <input type="submit" value="Mostrar alquileres cliente">
</form>
```

ClienteController

```
public ActionResult IndexAlquileresCliente()
        {
            return View("AlquileresCliente");
        }
```

```
public ActionResult ObtenerAlquileresCliente(int? idCliente)
        {
            if (idCliente != null) {
                ViewBag.Alquileres =
Sistema.Instancia.AlquileresCliente((int)idCliente);
            }
            return View("MostrarAlquileresCliente");
        }
```

View MostrarAlquileresCliente

```
@{
    ViewBag.Title =”MostrarAlquileresCliente”
}

@{

List<Alquiler> alquileresCliente = ViewBag.Alquileres;

    if (alquileresCliente.Count > 0) {
    <table>
        <th>Fecha de inicio</th>
        <th>Fecha de fin</th>
        <th>Oficina</th>
        @foreach (Alquiler miAlquiler in alquileresCliente) {
        <tr>
            <td>@miAlquiler.FechaInicio.ToString()</td>
            <td>@miAlquiler.FechaFin.ToString()</td>
            <td>@miAlquiler.Oficina</td>
        </tr>
        }
        </table>

    }
    else {
        <p>No existen alquileres para el cliente</p>
        }
    }
```