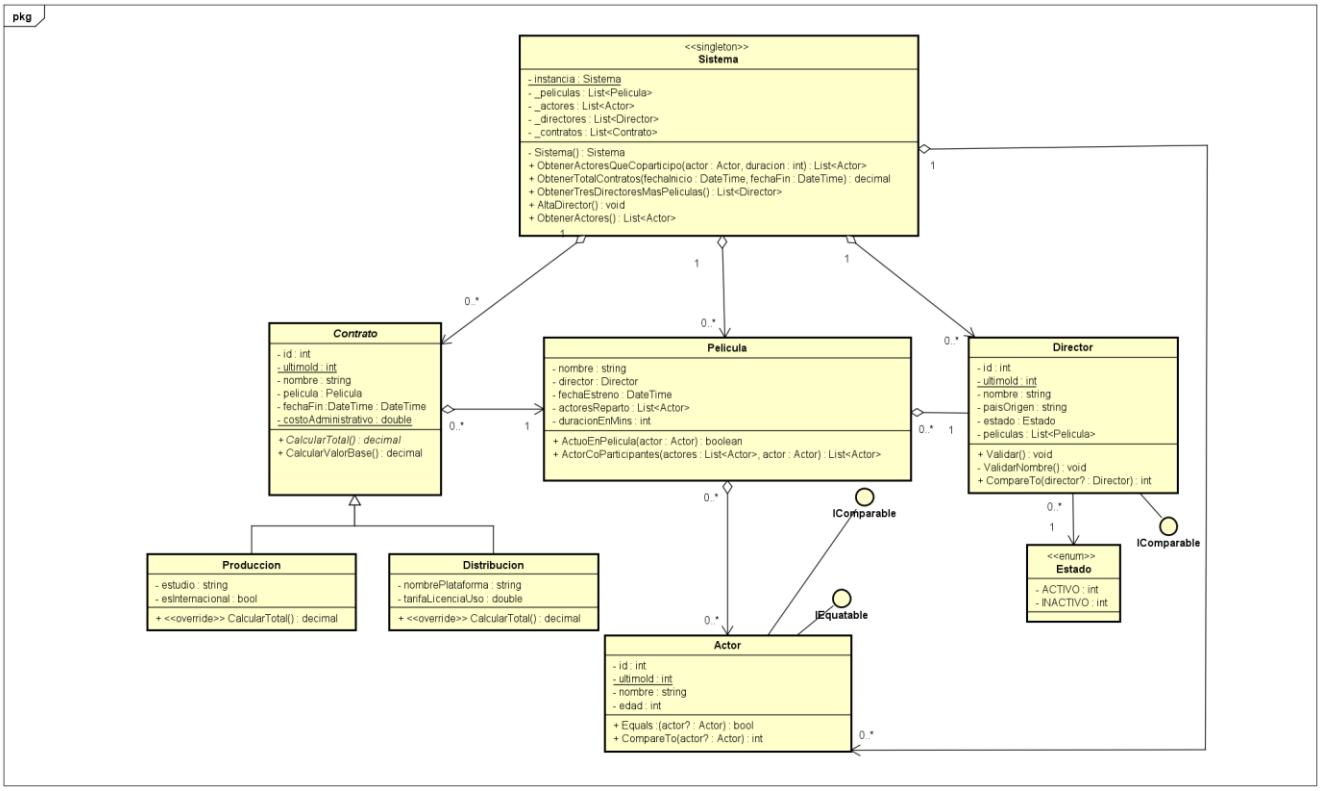
Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70
www.ort.edu.uy

EVALUACIÓN	EXAMEN	**GRUPO**	TODOS	**FECHA**	**23/07/2024**
MATERIA	PROGRAMACIÓN 2				
CARRERA	AP/ATI				
CONDICIONES	**- Duración 3 horas** **- Puntos: 100** **- Sin material** **- Otros**: **Indicar nombre del docente del curso en primera hoja del examen** **- Consultas exclusivamente sobre la letra**				

UML

Diagrama de clases del dominio de la aplicación que incluya las clases con sus atributos, relaciones, y métodos para resolver los requerimientos abajo detallados. (**30 puntos**)



Punto A

Dado un actor y una duración en minutos, obtener todos los actores con los que el actor ha coparticipado en las diferentes películas registradas en el sistema que superen la duración mencionada. Los actores resultantes no deben ser repetidos. Se debe devolver la información ordenada por nombre de actor ascendentemente. (**15 puntos**)

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70
www.ort.edu.uy

```
// Sistema

// Dado un actor y una duración en minutos,
// obtener todos los actores con los que el actor ha coparticipado en las diferentes películas registradas
// en el sistema que superen la duración mencionada.
// Los actores resultantes no deben ser repetidos.
// Se debe devolver la información ordenada por nombre de actor ascendentemente.

0 referencias
public List<Actor> ObtenerActoresQueCoparticipo(Actor actor, int duracion)
{
    List<Actor> auxActores = new List<Actor>();
    foreach (Pelicula unaPelicula in _peliculas)
    {
        if (unaPelicula.ActuoEnPelicula(actor) &&
            unaPelicula.Duracion > duracion)
        {
            auxActores =unaPelicula.ActorCoParticipantes(auxActores, actor);
        }
    }
    auxActores.Sort();
    return auxActores;
}
```

```
// Pelicula
1 referencia
public bool ActuoEnPelicula(Actor Actor)
{
    return _actores.Contains(Actor);
}

1 referencia
public List<Actor> ActorCoParticipantes(List<Actor> auxActores, Actor actor)
{
    foreach (Actor unActor in _actores)
    {
        if (!unActor.Equals(actor)  &&
            !auxActores.Contains(unActor))
        {
            auxActores.Add(unActor);
        }
    }
    return auxActores;
}
```

Punto B

Dado una fecha de comienzo y fecha de fin, obtener el valor total de contratos que tiene la empresa en ese rango de fechas. (**20 puntos**)

```
// Sistema

// Dado una fecha de comienzo y fecha de fin,
// obtener el valor total de contratos que tiene la empresa en ese rango de fechas.

0 referencias
public decimal ObtenerTotalContratos(DateTime fechaInicio, DateTime fechaFin)
{
    decimal total = 0;
    foreach (Contrato unContrato in _contratos)
    {
        if (unContrato.FechaFin >= fechaInicio &&
            unContrato.FechaFin <= fechaFin)
        {
            total += unContrato.CalcularTotal();
        }
    }
    return total;
}
```

```
// Contrato

2 referencias
public decimal CalcularValorBase()
{
    return CostoAdministrativo + CostoBase * Pelicula.Duracion;
}

3 referencias
public abstract decimal CalcularTotal();
```

```
0 referencias
public class Produccion : Contrato
{
    1 referencia
    public bool EsInternacional { get; set; }

    2 referencias
    public override decimal CalcularTotal()
    {
        decimal valor = base.CalcularValorBase();
        if (EsInternacional)
        {
            valor *= (decimal)1.15;
        }
        return valor;
    }
}
```

```
0 referencias
public class Distribucion : Contrato
{
    1 referencia
    public decimal TarifaLicenciaUso { get; set; }

    2 referencias
    public override decimal CalcularTotal()
    {
        return base.CalcularValorBase() + TarifaLicenciaUso;
    }
}
```

Punto C

Obtener los tres directores que tienen más películas realizadas. (**10 puntos**)

```
// Sistema

// Obtener los tres directores que tienen más películas realizadas. (10 puntos)
0 referencias
public List<Director> ObtenerTresDirectoresMasPeliculas()
{
    List<Director> auxRespuesta = new List<Director>();
    _directores.Sort();

    if (_directores.Count <= 3)
    {
        auxRespuesta = _directores;
    }
    else
    {
        for (int i = 0; i < 3; i++)
        {
            auxRespuesta.Add(_directores[i]);
        }
    }
    return auxRespuesta;
}
```

```
// Director

0 referencias
public int CompareTo(Director? director)
{
    if (director == null)
    {
        return -1;
    }
    return _peliculas.Count().CompareTo(director._peliculas.Count()) * -1;
}
```

MVC

Agregar un director al sistema solicitando todos sus datos. (**15 puntos**)

```
0 referencias
public class DirectorController : Controller
{
    private static Sistema _sistema = Sistema.Intancia;

    [HttpGet]
    0 referencias
    public IActionResult Alta()
    {
        return View(new Director());
    }

    [HttpPost]
    0 referencias
    public IActionResult Alta(Director director)

    {
        try
        {
            _sistema.AltaDirector(director);
            ViewBag.Mensaje = "Alta exitosa";
        }
        catch (Exception e)
        {
            ViewBag.Mensaje = e.Message;
        }
        return View(director);
    }
}
```

```
@using Dominio
@model Director

<h1>Alta de Director</h1>

@if (ViewBag.Mensaje != null)
{
    <p>@ViewBag.Mensaje</p>
}

<form method="post">
    <label>Nombre:</label>
    <input type="text" name="Nombre" value="@Model.Nombre" required />
    <label>PaisOrigen:</label>
    <input type="text" name="PaisOrigen" value="@Model.PaisOrigen" required />
    <select name="Estado">
        <option value="0">Activo</option>
        <option value="1">Inactivo</option>
    </select>
    <input type="submit" value="Ingresar" />
</form>
```

```
1 referencia
public void AltaDirector(Director director)
{
    if (director == null)
    {
        throw new Exception("No se recibieron datos para el director");
    }
    director.Validar();
    _directores.Add(director);
}
```

```
// Director
1 referencia
public void Validar()
{
    ValidarNombre();
}

1 referencia
private void ValidarNombre()
{
    if (string.IsNullOrEmpty(Nombre))
    {
        throw new Exception("No es válido el nombre");
    }
}
```

Listar todos los actores del sistema. Se deberá mostrar todos sus datos. (**10 puntos**)

```
0 referencias
public class ActorController : Controller
{
    private static Sistema _sistema = Sistema.Intancia;
    0 referencias
    public IActionResult Index()
    {
        return View(_sistema.Actores);
    }
}
```

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70
www.ort.edu.uy

```
@using Dominio
@model List<Actor>

<h1>Listado de Actores</h1>

@if (Model == null || Model.Count == 0)
{
    <p>No hay actores</p>
}
else
{
    <table>
        <thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col">First</th>
                <th scope="col">Last</th>
                <th scope="col">Handle</th>
            </tr>
        </thead>
        <tbody>
            @foreach (Actor item in Model)
            {
                <tr>
                    <td>@item.Id</td>
                    <td>@item.Nombre</td>
                    <td>@item.Edad</td>
                </tr>

            }
        </tbody>
    </table>
}
```