13019 - Diseño de bases de datos *Curso 2004-2005*
Fecha: *27 de enero de 2005*

Nombre: _____ Apellidos: _____

1. (*2.50 puntos*) La asociación de distribuidores cinematográficos de una determinada ciudad quiere crear un directorio en el que se pueda hacer consultas detalladas sobre las películas que se están proyectando en cualquiera de los cines de la ciudad:

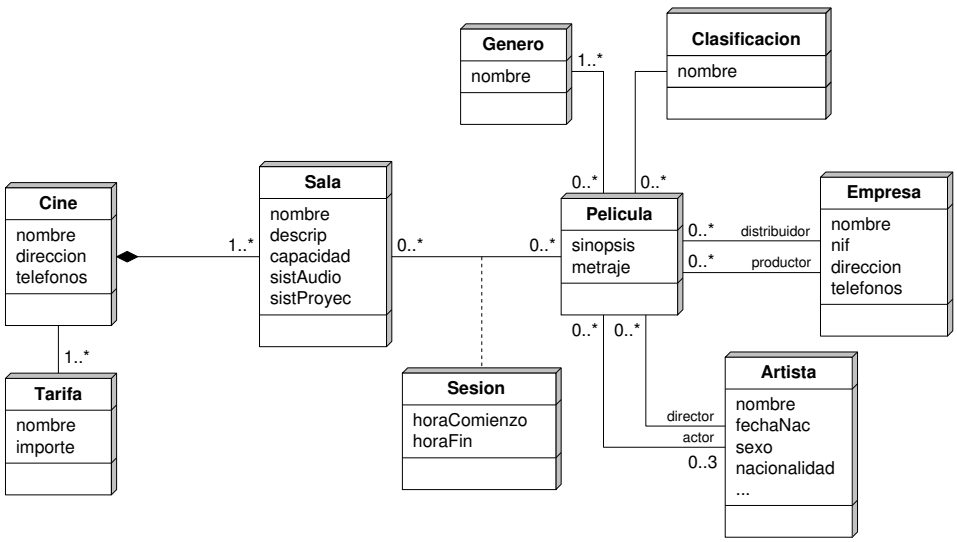
   - Detalles de una determinada película en proyección: director, hasta tres de los protagonistas, empresa productora, distribuidor, género o géneros[1] en que se clasifica y sinopsis argumental.
   - En qué cines se está proyectando una determinada película y el horario de los pases.
   - Qué películas de un determinado género se están proyectando y en qué cines.
   - Qué películas se están proyectando en un determinado cine en una franja horaria determinada.
   - Qué películas se están proyectando en la ciudad clasificadas por cines (cartelera): para cada cine se debe dar el nombre y la dirección del local, el título de la película o películas que se proyectan (es el caso de multicines, con indicación de las salas), el horario de los pases, los detalles de la película, los géneros a los que pertenece la película y la clasificación[2].

   La base de datos también deberá almacenar la calle y número donde se encuentra el cine, los teléfonos de contacto y el precio de la entrada según el día[3].

   De acuerdo con la descripción del sistema de información requerido:

   (a) (*0.50 puntos*) Describa qué se solicita del sistema mediante in diagrama de casos de uso.

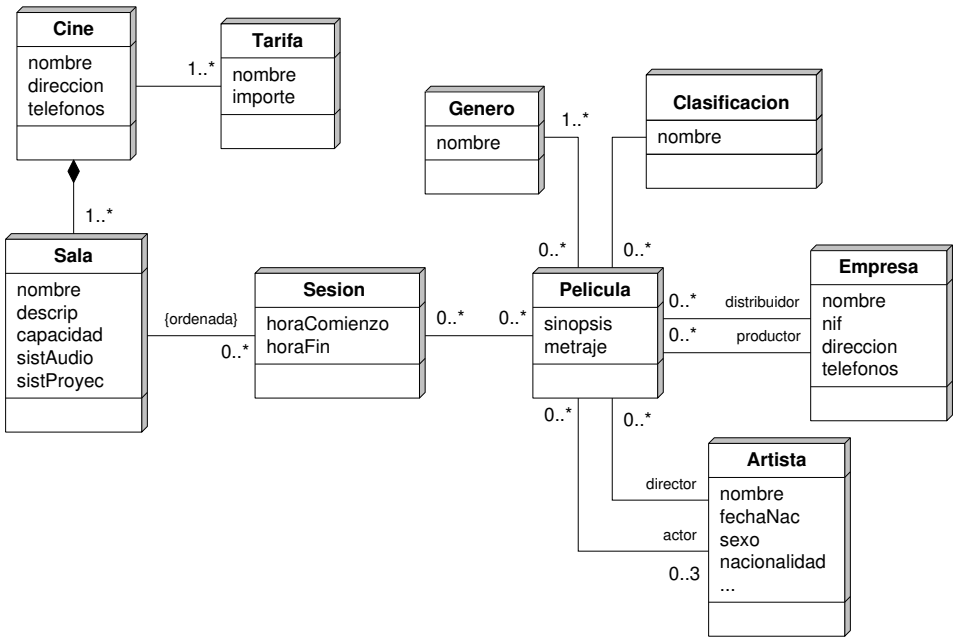
   (b) (*2.00 puntos*) Diseñe el diagrama de clases UML del sistema utilizando un nivel de detalle adecuado a la información proporcionada en el enunciado.



[1] Los géneros son: comedia, intriga, *western*, ciencia ficción, terror, *gore*, de Santiago Segura, drama, infantil, dibujos animados, animación 3D, etc...

[2] Las películas se clasifican en: *todos los públicos*, *mayores de 8 años*, *mayores de 13 años*, *mayores de 18 años* y *matriculados en DBD*.

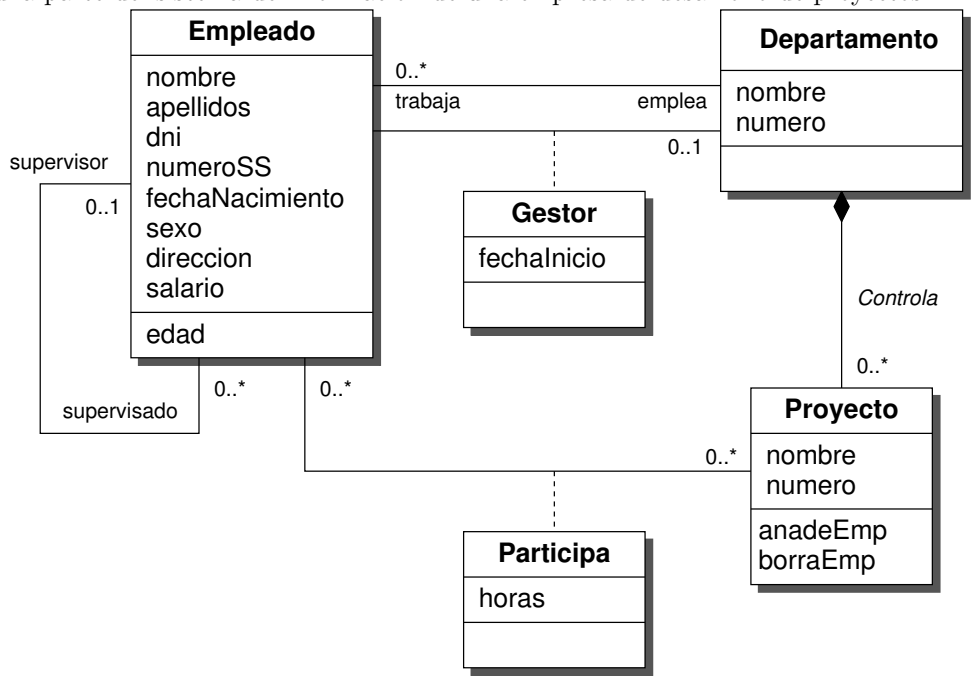
[3] Los días se clasifican en: *día del espectador*, *día del jubilado*, *festivos y vísperas de festivo*, *ordinario*, *carnet de estudiante*, etc...


Cine
nombre
direccion
telefonos
1..*
Tarifa
nombre
importe
1..*
Sala
nombre
descrip
capacidad
sistAudio
sistProyec
{ordenada}
0..*
Sesion
horaComienzo
horaFin
0..*
0..*
Genero
nombre
1..*
0..*
Clasificacion
nombre
0..*
Pelicula
sinopsis
metraje
0..*
distribuidor
0..*
productor
Empresa
nombre
nif
direccion
telefonos
0..*
0..*
director
actor
0..3
Artista
nombre
fechaNac
sexo
nacionalidad
...

13019 - Diseño de bases de datos *Curso 2004-2005*
Fecha: *27 de enero de 2005*

Nombre: _____ Apellidos: _____

2. (*2.50 puntos*) En la figura adjunta se muestra un diagrama de clases UML que representa una parte del sistema de información de una empresa de desarrollo de proyectos:



(a) (*1.50 puntos*) Proyecte el diagrama de clases en el modelo relacional. Recuerde que debe definir expresamente la identidad de los objetos del modelo.

(b) (*0.25 puntos*) Escriba una consulta SQL que muestre la lista (nombre, apellidos, dni, ...) de los empleados supervisados por un empleado dado identificado por su DNI.

(c) (*0.50 puntos*) Escriba una consulta SQL que muestre la lista de todos los empleados (nombre, apellidos, departamento, horas, ...) que participan en un proyecto dado ordenados por el número de horas dedicadas y por apellido.

(d) (*0.25 puntos*) Escriba una consulta SQL que muestre la dedicación a proyectos (nombre de proyecto, horas dedicadas) y la dedicación total (suma de todas las horas dedicadas) de un empleado dado identificado por su DNI.

(a)
```
CREATE TABLE Departamento (
  idDepto     NUMBER,
  nombre      VARCHAR2(60),
  PRIMARY KEY (idDepto)
 );

CREATE TABLE Empleado (
  idEmpl      NUMBER,
  nombre      VARCHAR2(30),
  apellidos   VARCHAR2(60),
  dni         CHAR(9),
  numeroSS    CHAR(15),
  fechaNac    DATE,
  sexo        CHAR(1),
  direccion   VARCHAR2(60),
```

```
    salario     NUMBER(12,2),
    supervisor NUMBER,
    idDepto       NUMBER,
    PRIMARY KEY (idEmpl),
    FOREIGN KEY (supervisor) REFERENCES Empleado
      ON DELETE SET NULL,
    FOREIGN KEY (idDepto) REFERENCES Departamento
      ON DELETE SET NULL
   );

  CREATE TABLE Gestor (
    idEmpl      NUMBER,
    idDepto     NUMBER,
    fechalnic   DATE,
    PRIMARY KEY (idDepto, idEmpl),
    FOREIGN KEY (idDepto) REFERENCES Departamento,
    FOREIGN KEY (idEmpl) REFERENCES Empleado
   );

  CREATE TABLE Proyecto (
    idProyec    NUMBER,
    idDepto     NUMBER,
    nombre      VARCHAR2(60),
    PRIMARY KEY (idProyec),
    FOREIGN KEY (idDepto) REFERENCES Departamento
      ON DELETE CASCADE
   );

  CREATE TABLE Participa (
    idProyec    NUMBER,
    idEmpl      NUMBER,
    horas       NUMBER,
    PRIMARY KEY (idProyec, idEmpl),
    FOREIGN KEY (idProyec) REFERENCES Proyecto,
    FOREIGN KEY (idEmpl) REFERENCES Empleado
   );
```

(b) (i)
```
SELECT e.nombre, e.apellidos, e.dni, d.nombre
  FROM Empleado e, Departament d
 WHERE e.supervisor = (
            SELECT e.idEmpl
              FROM Empleado e
             WHERE e.dni = '_UN_DNI_DADO_')
   AND e.idDepto = d.idDepto;
```

(ii)
```
SELECT e.nombre, e.apellidos, e.dni, d.nombre
  FROM Empleado e, Empleado s, Departament d
 WHERE s.dni = '_UN_DNI_DADO_'
   AND s.idEmpl = e.supervisor
   AND e.idDepto = d.idDepto;
```

(c)
```
SELECT e.nombre, e.apellidos, d.nombre, p.horas
  FROM Empleado e, Departamento d, Participa p
 WHERE p.idProyec = _UN_PROYECTO_DADO_
   AND p.idEmpl = e.idEmpl
   AND e.idDepto = d.idDepto
 ORDER BY (p.horas, e.apellidos);
```

(d) (i)
```
SELECT p.nombre, d.horas, SUM(d.horas)
  FROM Proyecto p, Participa d
 WHERE d.idEmpl = (
            SELECT e.idEmpl
              FROM Empleado e
             WHERE e.dni = '_UN_DNI_DADO_')
   AND d.idProyec = p.idProyec;
```

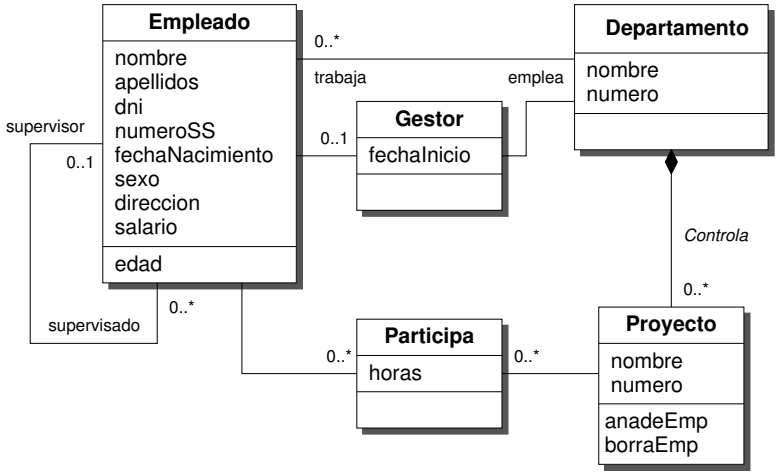
(ii)
```
SELECT p.nombre, d.horas, SUM(d.horas)
  FROM Empleado e, Proyecto p, Participa d
 WHERE e.dni = '_UN_DNI_DADO_'
   AND e.idEmpl = d.idEmpl
   AND d.idProyec = p.idProyec;
```

13019 - Diseño de bases de datos *Curso 2004-2005*
Fecha: *27 de enero de 2005*

Nombre: _____ Apellidos: _____

3. (*2.00 puntos*) Partiendo del diagrama de clases UML de la figura anterior, responda a las siguientes cuestiones:
   (a) (*0.50 puntos*) Modifique el diagrama de clases mediante una transformación de equivalencia promoviendo las clases de asociación a clases.
   (b) (*1.00 puntos*) Proyecte el nuevo diagrama de clases transformado en el modelo orientado a objetos de acuerdo con el estándar ODMG/C++. Todas las asociaciones deben considerarse bidireccionales.
   (c) (*0.25 puntos*) Implemente el método int Empleado::edad() const.
   (d) (*0.25 puntos*) Implemente el destructor de la clase Departamento.

(a) El diagrama transformado es:



(b) Una implementación es:

```
#include <odmg.h>

class Departamento : public d_Object {
private:
  d_String nombre;
  int numero;
protected:
  d_Rel_Set<Empleado,"trabaja"> emplea;
  d_Rel_Ref<Gestor,"departamento"> gestor;
  d_Rel_Set<Proyecto,"departamento"> proyectos;
public:
  ~Departamento();
};

class Empleado : public d_Object {
private:
  d_String nombre, apellidos, dni, numeroSS;
  d_Date fechaNacimiento;
  d_Char[2] sexo;
  d_String direccion;
```

```
  d_Float salario;
protected:
  d_Rel_Ref<Departamento,"emplea"> trabaja;
  d_Rel_Ref<Gestor,"empleado"> gestiona;
  d_Rel_Ref<Empleado,"supervisado"> supervisor;
  d_Rel_Set<Empleado,"supervisor"> supervisado;
  d_Rel_Set<Participa,"empleado"> proyectos;
public:
  int edad(void) const;
};

class Gestor : public d_Object {
private:
  d_Date fechaInicio;
protected:
  d_Rel_Ref<Departamento,"gestor"> departamento;
  d_Rel_Ref<Empleado,"gestiona"> empleado;
};

class Proyecto : public d_Object {
private:
  d_String nombre;
  int numero;
protected:
  d_Rel_Ref<Departamento,"proyectos"> departamento;
  d_Rel_Set<Participa,"proyecto"> empleados;
};

class Participa : public d_Object {
private:
  int horas;
protected:
  d_Rel_Ref<Proyecto,"empleados"> proyecto;
  d_Rel_Ref<Empleado,"proyectos"> empleado;
};
```

(b) Implemente el método int Empleado::edad() const:

```
int Empleado::edad(void) const {
  d_Date hoy; // fecha actual
  if (hoy.day_of_year() >= fechaNacimiento.day_of_year() )
    return hoy.year() - fechaNacimiento.year();
  else
    return hoy.year() - fechaNacimiento.year() - 1;
}
```

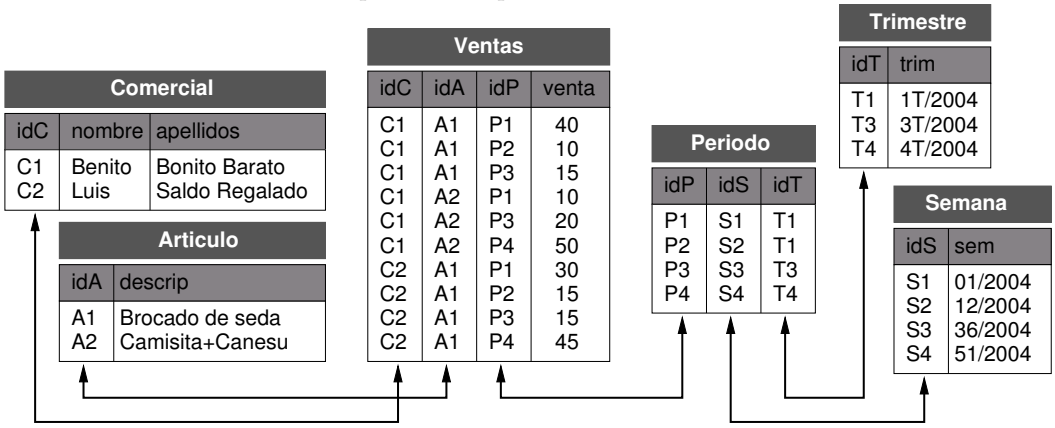
(c) Implemente el destructor de la clase Departamento:

```
Departamento::~Departamento() {
  // Borramos los proyectos asociados al departamento
  d_Iterator<d_Ref<Proyecto>> ip;
  d_Ref<Proyecto> p;
  for (ip = proyectos.create_iterator(); ip.not_done(); ip.advance()) {
     p = ip.get_element();
     p.delete_object();
  }
  // Borramos el objeto "Gestor"
  gestor.delete_object();
}
```

13019 - Diseño de bases de datos *Curso 2004-2005*
Fecha: *27 de enero de 2005*

Nombre: _____ Apellidos: _____

4. (*1.50 puntos*) En la figura siguiente se muestra un extracto de las tablas de un almacén de datos estructurado en un esquema en copo alrededor de la tabla de hechos Ventas:



Complete los datos que faltan en la salida generada por las siguientes consultas ROLAP:

(a) (*0.50 puntos*)

```
SELECT c.nombre, c.apellidos, a.descrip, SUM(v.venta)
  FROM Comercial c, Articulo a, Ventas v, Periodo p
 WHERE p.idT='T3' AND p.idP=v.idP AND c.idC=v.idC and a.idA=v.idA
 GROUP BY CUBE ((c.nombre, c.apellidos), a.descrip);
```

NOMBRE	APELLIDOS	DESCRIP	SUM(V.VENTA)
Benito	Bonito Barato	Brocado de seda	15
Benito	**Bonito Barato**	**Camisita+Canesu**	**20**
Luis	**Saldo Regalado**	Brocado de seda	**15**
Benito	Bonito Barato	*Null*	35
Luis	**Saldo Regalado**	*Nulll*	**15**
Null	*Null*	**Brocado de seda**	30
Null	*Null*	Camisita+Canesu	**20**
Null	*Null*	*Null*	**50**

(b) (*1.00 puntos*)

```
SELECT c.apellidos, a.descrip, SUM(v.venta)
  FROM Comercial c, Articulo a, Ventas v, Periodo p
 WHERE p.idT='T1' AND p.idP=v.idP AND c.idC=v.idC and a.idA=v.idA
 GROUP BY ROLLUP (c.apellidos, a.descrip);
```

APELLIDOS	DESCRIP	SUM(V.VENTA)
Bonito Barato	**Brocado de seda**	**50**
Bonito Barato	**Camisita+Canesu**	**10**
Bonito Barato	**Null**	**60**
Saldo Regalado	**Brocado de seda**	**45**
Saldo Regalado	**Null**	**45**
Null	*Null*	105

13019 - Diseño de bases de datos *Curso 2004-2005*
Fecha: *27 de enero de 2005*

Nombre: _____ Apellidos: _____

5. (*1.50 puntos*) Modelo objeto-relacional de Oracle: necesidad, uso y definición de los métodos MAP y ORDER en la construcción de tipos. Ilustre el desarrollo de la pregunta utilizando varios ejemplos.