13019 - Diseño de bases de datos *Curso 2005-2006*
Fecha: *2 de febrero de 2006*

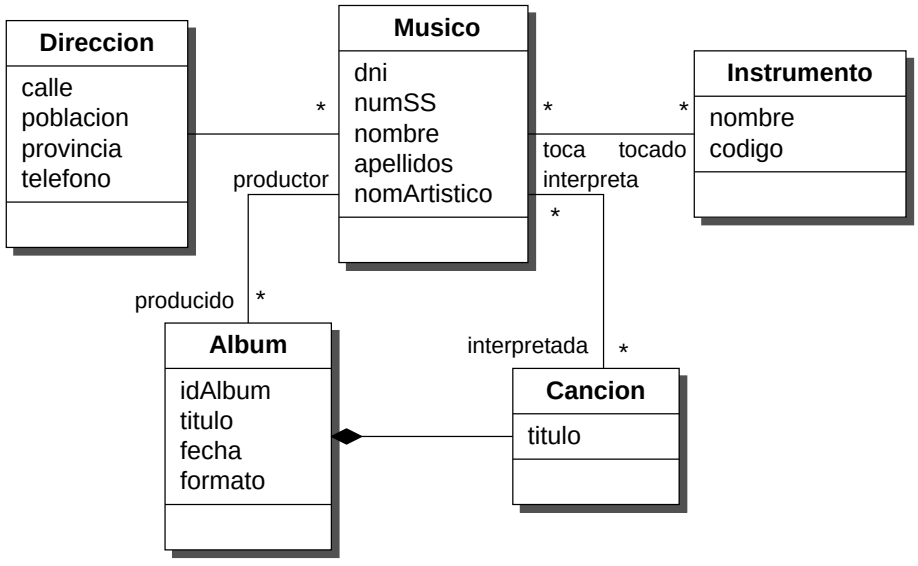
Nombre: _____ Apellidos: _____

1. (*2.00 puntos*) La empresa discográfica **19CHULOS** (más conocida en el ambiente musical como *Nineteen Cockyes Records*) ha decidido almacenar la información relativa a los músicos que participan en sus discos (junto con más información útil para la compañía) en una base de datos. La empresa ha tomado la sabia y prudente decisión de contratarte como diseñador de bases de datos (a la tarifa habitual para un consultor de 2.040€/día). Los requisitos del sistema son:

   - Cada músico que graba en **19CHULOS** tiene DNI, número de la Seguridad Social, nombre y apellidos, nombre artístico, dirección y un número de teléfono. Los músicos están generalmente mal pagados, por lo que es frecuente que compartan la residencia y ninguna dirección cuenta con más de un teléfono.
   - Todos los instrumentos utilizados en las grabaciones de **19CHULOS** están registrados y tienen asociado un nombre (por ejemplo *guitarra*, *sintetizador*, *flauta*, etc...) y un código de instrumento.
   - Cada *Album* que es grabado en la compañía tiene un título, una fecha de *copyright*, un determinado formato (p.e., CD, DVD, MC) y un identificador.
   - Cada músico puede tocar varios instrumentos y cada instrumento puede ser tocado por varios músicos.
   - Cada *Album* cuenta con un cierto número de canciones, pero una canción no puede aparecer en más de un *Album*.
   - Cada canción es interpretada por uno o más músicos y cada músico puede participar en más de una canción.
   - Cada *Album* está exactamente asociado a un músico en el papel de productor. Por supuesto, un músico puede producir varios albumes.

   Teniendo en cuenta la información anterior, diseñe un diagrama de clases UML que modele el sistema solicitado por la empresa.

La solución es:



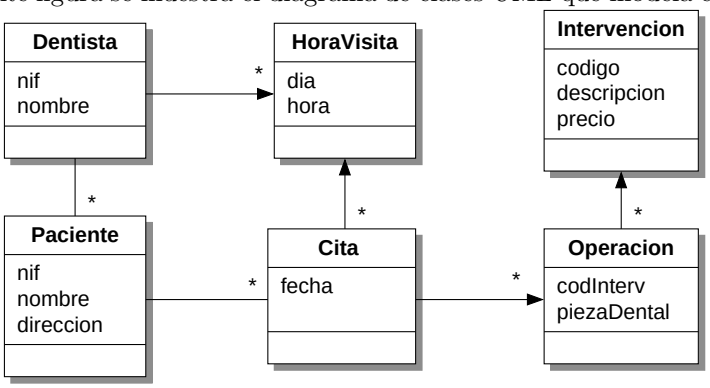
13019 - Diseño de bases de datos *Curso 2005-2006*
Fecha: *2 de febrero de 2006*

Nombre: _____ Apellidos: _____

2. (*3.00 puntos*) Una clínica dentista desea informatizar parte de la gestión de su actividad. En el consultorio trabaja un equipo de dentistas en las siguientes condiciones:

   - Cada dentista es responsable único del tratamiento de sus pacientes y cada paciente es tratado siempre por el mismo dentista.
   - Cada dentista trabaja un conjunto de horas al día y programa las visitas para sus pacientes en estas horas. Las visitas se distribuyen en fracciones de 15 minutos y son fijas para cada semana, por ejemplo (Lunes,8:15),(Lunes,8:30),(Jueves,12:00), etc...
   - Las operaciones realizadas se codifican mediante un código interno de dos dígitos, de modo que el primer dígito describe la intervención (por ejemplo, 112 para extracción, 101 para desnaturalización, 48 para ortodoncia, etc..) y el segundo la numeración de la pieza afectada (por ejemplo, 3 para el primer molar superior izquierdo, 15 para el premolar inferior derecho, etc...).
   - Cada intervención tiene un precio asociado que es independiente de la pieza dental afectada y que se utiliza para calcular la facturación del paciente. La clínica factura a sus clientes con periodicidad mensual.
   - En una visita se pueden realizar varias operaciones.

   En la siguiente figura se muestra el diagrama de clases UML que modela el sistema descrito:



   (a) (*0.25 puntos*) Defina expresamente la identidad de los objetos del modelo.

   (b) (*0.25 puntos*) Construya el enumerado piezaDental.

   (c) (*1.75 puntos*) Proyecte el diagrama de clases en el modelo relacional.

   (d) (*0.25 puntos*) Dado un paciente identificado por su nif, construya una consulta SQL que proporcione la lista ordenada de todas las citas que ha realizado a su dentista. El resultado de la consulta debe mostrar la fecha, el día y la hora de la cita.

   (e) (*0.50 puntos*) Escriba una consulta SQL que, dado el nif de un cliente y un número de mes, emita la factura del cliente en el mes indicado. El resultado de esa consulta debe indicar: la fecha y codificación de la operación, la descripción de la intervención, el nombre de la pieza afectada y el importe asociado.

(a) Una posibilidad, que utilizaremos en el resto del desarrollo, es:

Dentista(idDentista, nif, nombre)
HoraVisita(idHoraVisita, dia, hora)
Paciente(idPaciente, nif, nombre, direccion)
Cita(idCita, fecha)
Operacion(idOperacion, codInterv, piezaDental)
Intervencion(codigo, descripcion, precio)

(b) **PiezaDental**(codigoPieza, nombre)

(c) La proyección en el esquema relacional es:

```
 CREATE TABLE Dentista (
   idDentista INTEGER PRIMARY KEY,
   nif        VARCHAR(20),
   nombre     VARCHAR(50)
);

CREATE TABLE HoraVisita (
  idHoraVisita INTEGER PRIMARY KEY,
  dia          INTEGER,
  hora         INTEGER,
  idDentista   INTEGER,
  FOREIGN KEY (dentista) REFERENCES Dentista
);

CREATE TABLE Paciente (
  idPaciente INTEGER PRIMARY KEY,
  nif        VARCHAR(20),
  nombre     VARCHAR(50),
  direccion  VARCHAR(100),
  dentista INTEGER,
  FOREIGN KEY (dentista) REFERENCES Dentista
);

CREATE TABLE Cita (
  idCita   INTEGER PRIMARY KEY,
  fecha    DATE,
  visita   INTEGER,
  paciente INTEGER,
  FOREIGN KEY (visita) REFERENCES HoraVisita,
  FOREIGN KEY (paciente) REFERENCES Paciente
);

CREATE TABLE Intervencion (
  codigo      INTEGER PRIMARY KEY,
  descripcion VARCHAR(50),
  precio      FLOAT
);

CREATE TABLE PiezaDental (
  codigoPieza INTEGER PRIMARY KEY,
  nombre      VARCHAR(40)
);

CREATE TABLE Operacion (
  idOPeracion INTEGER PRIMARY KEY,
  cita        INTEGER,
  codInterv   INTEGER,
  piezaDental INTEGER,
  FOREIGN KEY (cita) REFERENCES Cita,
  FOREIGN KEY (codInterv) REFERENCES Intervencion,
  FOREIGN KEY (piezaDental) REFERENCES PiezaDental
);
```

(d) Lista ordenada de todas las citas de un paciente a su dentista:

```
SELECT p.nombre, c.fecha, h.dia, h.hora
  FROM Paciente p, Cita c, HoraVisita h
 WHERE p.nif = _el_nif_de_un_paciente_
   AND p.idPaciente = c.paciente
   AND c.visita = h.idHoraVisita
ORDER BY c.fecha;
```

(e) Factura de un cliente en un mes indicado:

```
SELECT p.nombre, c.fecha, o.intervencion, i.descripcion,
       d.nombre, i.precio
  FROM Paciente p, Cita c, Operacion o, Intervencion i,
       PiezaDental d
 WHERE p.nif = _el_nif_de_un_paciente_
   AND p.idPaciente = c.paciente
   AND MONTH(c.fecha) = _un_numero_de_mes_
   AND c.idCita = o.cita
   AND o.codInterv = i.codigo
   AND o.piezaDental = d.codigoPieza
ORDER BY c.fecha;
```

13019 - Diseño de bases de datos *Curso 2005-2006*
Fecha: *2 de febrero de 2006*

Nombre: _____ Apellidos: _____

3. (*1.50 puntos*) Dadas las siguientes definiciones de tipos y tablas de Oracle:

```
CREATE TYPE Contraindicacion AS OBJECT (
  codigo INTEGER,
  descripcion VARCHAR2(250)
);

CREATE TYPE TablaContraindicaciones AS TABLE OF Contraindicacion;

CREATE TYPE Farmaco AS OBJECT (
  codigo INTEGER,
  nombre VARCHAR2(100),
  descripcion VARCHAR2(1200),
  formulaQuimica VARCHAR2(100),
  efectosNocivos TablaContraindicaciones
);

CREATE TABLE Farmacos OF Farmaco (
  PRIMARY KEY (codigo)
)
NESTED TABLE efectosNocivos STORE AS Tabla_de_Contraindicaciones;
```

(a) (*1.00 puntos*) Construya las sentencias SQL necesarias para introducir en la tabla Farmaco los datos relativos al principio activo de la conocida Aspirina:

Código	12
Nombre	Ácido acetil-salicílico
descripción	Analgésico, antiinflamatorio, antifebril, antigripal y antiagregante plaquetario.
Fórmula química	$C_9H_8O_4$
Contraidicaciones:	Transtornos de la coagulación. Asma. Ulcera gastroduodenal. Antecedentes de nefropatía. Embarazo, especialmente durante el último trimestre.

(b) (*0.50 puntos*) Escriba una consulta SQL que muestre la descripción de todas las contraindicaciones del ácido acetil-salicílico.

(a) Las sentencias de inserción son:

```
INSERT INTO Farmacos
VALUES (12,
        'Ácido acetil-salicílico',
        'Analgésico, antiinflamatorio, antifebril, ...',
        'C9H8O4',
        TablaContraindicaciones());

INSERT INTO THE (SELECT efectosNocivos
                   FROM Farmacos f
                  WHERE f.codigo = 12)
VALUES (1, 'Transtornos de la coagulación');

INSERT INTO THE (SELECT efectosNocivos
                   FROM Farmacos f
                  WHERE f.codigo = 12)
VALUES (2, 'Asma');
```

```
INSERT INTO THE (SELECT efectosNocivos
                   FROM Farmacos f
                  WHERE f.codigo = 12)
VALUES (3, 'Ulcera gastroduodenal');

INSERT INTO THE (SELECT efectosNocivos
                   FROM Farmacos f
                  WHERE f.codigo = 12)
VALUES (4, 'Antecedentes de nefropatía');

INSERT INTO THE (SELECT efectosNocivos
                   FROM Farmacos f
                  WHERE f.codigo = 12)
VALUES (5, 'Embarazo, especialmente durante...');
```

(b) Descripción de todas las contraindicaciones del ácido acetil-salicílico.

```
SELECT descripcion
  FROM THE (SELECT efectosNocivos
              FROM Farmacos f
             WHERE f.codigo = 12);
```

13019 - Diseño de bases de datos *Curso 2005-2006*
Fecha: *2 de febrero de 2006*

Nombre: _____ Apellidos: _____

ATENCION: Elija **una** de las siguientes cuestiones y conteste **sólo** a la elegida.

4. **[ODMG/C++]** (*2.00 puntos*) Partiendo del diagrama de clases UML de la figura anterior, responda a las siguientes cuestiones:

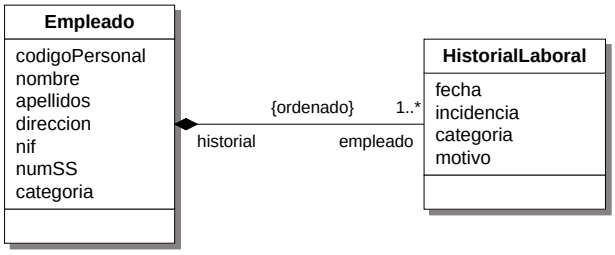
   (a) (*1.50 puntos*) Proyecte el diagrama de clases en el modelo orientado a objetos de acuerdo con el estándar ODMG/C++.

   (b) (*0.50 puntos*) Implemente el método float Paciente::factura(int mes) const que dado un número de mes devuelva el importe de todas las intervenciones realizadas a un paciente.

No se incluye la solución.

4. **[JDO]** (*2.00 puntos*) Responda a las siguientes cuestiones:

   (a) (*1.00 puntos*) Dado el diagrama de clases UML:



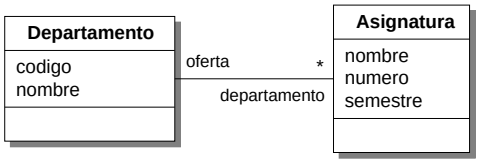
El siguiente código es erroneo:

```
public class Empleado {
  private int codigoPersonal;
  private String nombre;
  private String apellidos;
  private String dirección;
  private String nif;
  private String numSS;
  private Categoria categoria;
  protected Collection<HistorialLaboral> historial =
     new HashSet<HistorialLaboral>();
  ...
}
```

```
public class HistorialLaboral {
  private Calendar fecha;
  private String incidencia;
  private Categoria categoria;
  private String motivo;
  ...
}
```

Explique **por qué** y corrija el/los errores.

   (b) (*1.00 puntos*) Dado el siguiente diagrama de clases UML:



¿Es correcto el correspondiente fichero package.jdo?

```
<?xml version="1.0"?>
<!DOCTYPE jdo PUBLIC
  "-//Sun Microsystems, Inc.//DTD Java Data Objects Metadata 2.0//EN"
  "http://java.sun.com/dtd/jdo_2_0.dtd">
```

```
<jdo>
  <package name="....">
    <!-- Clase Departamento -->
    <class name="Departamento" identity-type="datastore">
      <inheritance strategy="new-table"/>
      <field name="codigo" persistence-modifier="persistent"/>
      <field name="nombre" persistence-modifier="persistent"/>
      <field name="oferta" persistence-modifier="persistent">
        <collection element-type="Asignatura"/>
      </field>
    </class>
    <!-- Clase Asignatura -->
    <class name="Asignatura" identity-type="datastore">
      <inheritance strategy="new-table"/>
      <field name="nombre" persistence-modifier="persistent"/>
      <field name="numero" persistence-modifier="persistent"/>
      <field name="semestre" persistence-modifier="persistent"/>
    </class>
  </package>
</jdo>
```

Conteste **si/no** y explique **por qué**.

(a) El código es erróneo por dos motivos:

1. El atributo `historial` de la clase `Empleado` que implementa la asociación con la clase `HistorialLaboral` se define como un conjunto (`Set`) cuando la asociación es **ordenada**. El código correcto es:

```
protected Collection<HistorialLaboral> historial =
   new ArrayList<HistorialLaboral>();
```

2. En la clase `HistorialLaboral` falta el atributo que implementa la asociación bidireccional con la clase `Empleado`. El código sería:

```
public class HistorialLaboral {
  private Calendar fecha;
  private String incidencia;
  private Categoria categoria;
  private String motivo;
  protected Empleado empleado;
}
```

(b) Hay también dos errores:

1. En la descripción del campo `oferta` de la clase `Departamento` falta especificar qué variable de la clase asociada implementa la relación inversa (`mapped-by=...`). El código quedaría:

```
<!-- Clase Departamento -->
<class name="Departamento" identity-type="datastore">
  <inheritance strategy="new-table"/>
  <field name="codigo" persistence-modifier="persistent"/>
  <field name="nombre" persistence-modifier="persistent"/>
  <field name="oferta"
     persistence-modifier="persistent"
    mapped-by="departamento" />
    <collection element-type="Asignatura"/>
  </field>
</class>
```

2. En la descripción de la clase `Asignatura` falta la especificar la variable `departamento` que implementa la asociación bidireccional:

```
<!-- Clase Asignatura -->
<class name="Asignatura" identity-type="datastore">
  <inheritance strategy="new-table"/>
  <field name="nombre" persistence-modifier="persistent"/>
  <field name="numero" persistence-modifier="persistent"/>
  <field name="semestre" persistence-modifier="persistent"/>
  <field name="departamento"
      persistence-modifier="persistent"
     mapped-by="oferta" />
</class>
```

13019 - Diseño de bases de datos *Curso 2005-2006*
Fecha: *2 de febrero de 2006*

Nombre: _____ Apellidos: _____

5. (*1.50 puntos*) Almacenes de datos: Procesamiento analítico en línea (GROUPING SETS, ROLLUP y CUBE).