**EXAMEN 2ª CONVOCATORIA**
Asignatura **Metodología y Tecnología de la Programación**
Fecha 25 de Junio de 2009 Curso: **2008/2009**

UNIVERSITAT DE BARCELONA

**Notas importantes**:
- El examen se debe hacer en **bolígrafo** y hay que poner **el nombre, el NIUB y el número de página** en todas las hojas que entreguéis.
- Se deben utilizar colecciones.
- Se deben explicar todos los aspectos del diseño que no queden claros. No basta con los diagramas.
- El examen es incremental, se deben realizar los apartados en el orden solicitado.

**Puntuación**: 70 % del total del examen **Tiempo estimado**: 3 ½ horas

**La parte práctica debe ser igual o superior a 4 para hacer media con la parte teórica**

## Problema 1 Clínica veterinaria (ClinicVet)

La clínica veterinaria quiere gestionar los clientes y las mascotas de sus clientes. Un cliente es un conjunto de personas que suele corresponderse con una familia. Cada cliente tiene un código que lo identifica, un número de cuenta bancaria, un teléfono y los nombres y DNI de las personas correspondientes. No existe límite en el número de personas asociadas a un cliente. Además, una persona puede estar dada de alta en varios clientes (por ejemplo, un hombre que vive con su esposa tiene un gato y como tal pertenece a un cliente, pero también esta dado de alta en el cliente asociado con el perro de sus padres).

Un cliente a su vez puede tener varias mascotas, cada mascota tiene un identificador, un nombre, la raza y su fecha de nacimiento. Cada raza guarda la información del nombre de la raza y las características propias de cada raza. Por ejemplo, para los perros se guardará el color del pelo, en cambio para los pájaros el tipo de plumaje y la forma del pico. De cada mascota se quiere guardar su historial médico, el cual guardarà todas la enfermedades que ha sufrido y todas las vacunas que se le han puesto. De la enfermedad se sabe el nombre y el tratamiento que se le ha dado. De las vacunas se sabe, la enfermedad que previene, el nombre de la vacuna y el precio de la misma. Además del historial médico, se quiere conocer todas las visitas que ha hecho la mascota a la clínica veterinaria. Estas visitas guardan, el precio de la visita, la fecha de la visita, el peso de la mascota, si ha sido pagada y las vacunas que se le han puesto (en una visita es posible que no se pongan vacunas o que se pongan más de una).

La aplicación a diseñar deberá asumir que la introducción de los datos ya se ha realizado. El objetivo de la aplicación es permitir añadir nuevas mascotas, añadir nuevos clientes asociándolos con su mascota, cambiar de dueño una mascota, añadir personas a un cliente y centrarse en la gestión de diversas consultas.

## Se solicita:

1. **(2 puntos)** Construir el modelo de dominio utilizando UML y completándolo con una descripción detallada que facilite su comprensión. En el modelo de dominio, los roles de las asociaciones deben contener el nombre, la multiplicidad y la dirección de lectura.
2. **(6.5 puntos)** Diseñar en detalle los eventos descritos a continuación utilizando diagramas de secuencia. Para cada uno de los eventos de sistema de este apartado, hay que detallar claramente sus parámetros y describir qué patrones usáis y porqué.

**Importante**: Intentad repartir al máximo las responsabilidades en los diagramas. Se evaluaran negativamente las clases "dios".

**a.** Mostrar para cada cliente el coste de todas las visitas no pagadas realizadas a sus mascotas y la cuenta bancaria donde se tiene que cobrar. El coste de cada visita debe incluir también el coste de las vacunas. Si una visita ha sido pagada, también se han pagado las vacunas que se hayan podido poner.

**b.** Dado el DNI de una persona y el nombre de una raza, mostrar el número total de mascotas que esa persona tiene de esa raza.

**c.** Realizar el cambio de cliente de una mascota. Para ello nos proporcionan el identificador de cliente del actual dueño, el identificador de cliente del nuevo dueño y el nombre de la mascota.

**Consideraciones del apartado c**:

- En el caso de que el dueño actual no tenga más mascotas, éste se eliminará de nuestra clínica veterinaria. Si por el contrario, tiene más mascotas, sólo hay que eliminar la mascota de ese cliente.
- La mascota se tiene que asociar con el nuevo dueño.

3. **(1.5 puntos)** Realizar el diagrama de clases (*incluid los setters i getters que uséis*) del diseño realizado en el **apartado 2**.

## Interface Collection

| Method Summary | |
|---|---|
| boolean | **add**(Object o)<br>Ensures that this collection contains the specified element (optional operation). |
| boolean | **addAll**(Collection c)<br>Adds all of the elements in the specified collection to this collection (optional operation). |
| void | **clear**()<br>Removes all of the elements from this collection (optional operation). |
| boolean | **contains**(Object o)<br>Returns true if this collection contains the specified element. |
| boolean | **containsAll**(Collection c)<br>Returns true if this collection contains all of the elements in the specified collection. |
| boolean | **equals**(Object o)<br>Compares the specified object with this collection for equality. |
| int | **hashCode**()<br>Returns the hash code value for this collection. |
| boolean | **isEmpty**()<br>Returns true if this collection contains no elements. |
| Iterator | **iterator**()<br>Returns an iterator over the elements in this collection. |
| boolean | **remove**(Object o)<br>Removes a single instance of the specified element from this collection, if it is present (optional operation). |
| boolean | **removeAll**(Collection c)<br>Removes all this collection's elements that are also contained in the specified collection (optional operation). |
| boolean | **retainAll**(Collection c)<br>Retains only the elements in this collection that are contained in the specified collection (optional operation). |
| int | **size**()<br>Returns the number of elements in this collection. |
| Object[] | **toArray**()<br>Returns an array containing all of the elements in this collection. |
| Object[] | **toArray**(Object[] a)<br>Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array. |