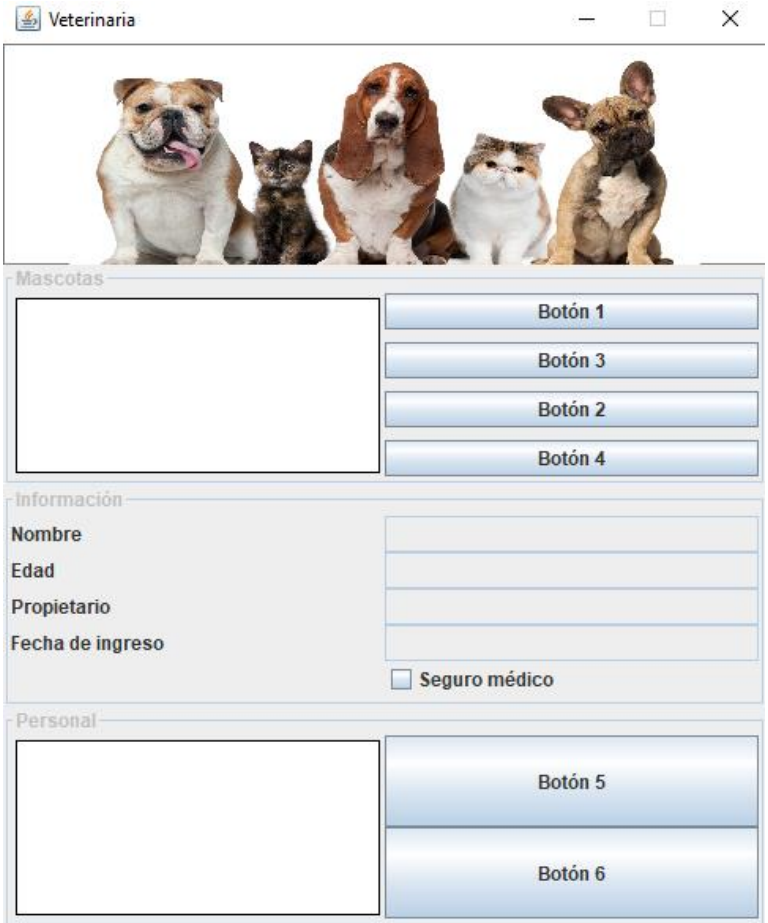
Enunciado:

En una clínica veterinaria, se requiere una aplicación que permita tener el registro de los animales que están internados por alguna situación médica, y que permita reconocer a las personas que se encuentran en turno de atención (la veterinaria funciona 24 horas). Para este fin, se tiene en cuenta que para cada animal que ingresa se tiene un nombre, el nombre del propietario, un número de contacto, fecha de ingreso, edad, y se puede solicitar alguna información extra si se necesita. De las personas que están en servicio, se conoce su nombre, su documento de identificación, el horario que tienen (6am -2pm, 2pm-10pm, 10pm-6am) y también se puede pedir cierta información adicional si se requiere (como su puesto en la clínica). Se desea también poder registrar a los animales nuevos que ingresan.

La información inicial del sistema se carga de dos archivos externos (uno con los datos de las mascotas y otro con los datos del personal). Se desea mostrar una lista que contenga los datos de mascotas y otra donde se encuentren los datos del personal. También se desea adicionar un animal cuando llegue a la clínica y poder registrarlo en el sistema.

Se sugiere una interfaz como se muestra a continuación (Aunque se pueden hacer modificaciones si es necesario):



COMPRENSIÓN Y ESPECIFICACIÓN DEL PROBLEMA[1]

En la solución de un problema, usualmente se involucran tres pasos:

1. El diseño de la solución
2. El programa
3. Las pruebas de corrección del programa.

En este espacio, se trabajará lo correspondiente al diseño de la solución y se implementarán las herramientas aprendidas hasta hoy en la construcción del programa.

Para este fin, es necesario comprender cómo se diseña la solución de un problema. Entre los pasos que se siguen están los siguientes:

1. Identificar los requerimientos funcionales. Un requerimiento funcional es una operación que el programa que se va a construir debe proveer al usuario, y que está directamente relacionada con el problema que se quiere resolver. Un requerimiento funcional se describe a través de cuatro elementos:
   - Un identificador y un nombre. (En lo posible usar verbos para indicar el nombre). Esto se puede ver reflejado al final como la **signatura** de los métodos.
   - Un resumen de la operación que debe realizar el requerimiento.
   - Las entradas (datos) que debe dar el usuario para que el programa pueda realizar la operación. Puede haber cero, una o más entradas para un requerimiento. Usualmente estas se ven reflejadas en los **parámetros** de los métodos.
   - El resultado esperado de la operación. Hay tres tipos posibles de resultados de un requerimiento funcional: (1) Una modificación de un valor en el mundo del problema, (2) el cálculo de un valor, o (3) una mezcla de los dos anteriores.

   No siempre estos requerimientos son reconocidos por el usuario. A veces, es tarea del programador indicarlos y añadirlos como posibles requerimientos al problema. Usualmente estos requerimientos suelen verse representados en los **métodos** de las clases.

   Ejemplo: (Para el problema del cajón de figuras)

   Nombre: (R1) Determinar el número de figuras con perímetro mayor a un valor dado.

   Resumen: Busca en la lista de figuras las figuras con mayor perímetro de acuerdo con valor dado y las cuenta.

   Entradas: Valor de referencia para comparar el perímetro de cada figura.

   Resultado: Retorna el número de figuras que tienen un perímetro mayor al valor dado. Si no hay figuras, debe retornar un mensaje que indique que no hay figuras.

[1] La mayoría de la información aquí presentada se tomó del libro *Fundamentos de Programación. Aprendizaje Activo Basado en Casos* de Jorge A. Villalobos y Rubby Casallas.

2. **Describir el modelo del mundo del problema.** Este paso consiste en identificar los principales elementos del mundo que participan en el problema, con sus principales características y relaciones. En palabras más sencillas, lo que se busca es encontrar cuáles son las clases principales del problema, sus atributos y cómo se relacionan entre ellas.
   Para este fin, se suelen identificar 4 partes importantes:
   - Identificar las entidades. Estos son los elementos que intervienen en el problema. Por ejemplo, cuando se está pensando en un problema que maneja figuras se pueden distinguir elementos como los puntos, los lados, las figuras, entre otros. En POO, llamamos a estas entidades las **clases.**
   - Modelar las características de las entidades. Cada una de estas características suele tener un nombre significativo y una descripción de los valores que puede tomar. Por ejemplo, si modelamos una persona, una característica que puede tener es el nombre. Y este nombre suele ser una cadena de caracteres; también se puede tener el salario (si es un empleado) que tiene que tomar valores positivos y expresados en alguna moneda. En POO, se suelen llamar a estas características los **atributos.**
   - Buscar las relaciones entre las entidades. Usualmente, las entidades del mundo suelen estar relacionadas directa o indirectamente, y se pueden mostrar gráficamente en un diagrama UML. Se suelen usar flechas para mostrar la relación, aunque existen modelos más sofisticados con indicaciones con flechas de distintas características. Un ejemplo de una relación o asociación entre dos entidades puede ser la que existe entre la clase Figura y la clase Lado. Cada figura tiene un número de lados, entonces existe una relación entre estas dos clases. Una asociación se puede ver como una característica de una entidad cuyo valor está representado por otra clase.
   - Documentar el resultado obtenido. NO OLVIDE DOCUMENTAR. Es importante mostrar al usuario un manual de cómo funciona el programa y esta es una herramienta invaluable para este fin.

3. **Identificar los requerimientos NO funcionales.** Usualmente al diseñar un programa es necesario reconocer las restricciones que tiene el usuario y las restricciones dadas por el contexto de utilización del programa. Usualmente las limitaciones están relacionadas con restricciones sobre la tecnología que se debe usar, el volumen de datos que se debe manejar o la cantidad de usuarios. Por ejemplo, no es lo mismo un programa que debe trabajar con un único usuario que uno que debe funcionar con miles simultáneamente.
   En el contexto que estamos trabajando, no es necesario considerar requerimientos no funcionales aparte de la visualización e interacción con la interfaz gráfica. Así que no son necesarios en este problema.

Luego de plantear una solución con las entidades y los requerimientos funcionales y no funcionales, se tiene que planear quién hace qué. Usualmente a esto se le llama la técnica del experto que, en breves palabras, busca determinar qué entidad es dueña de la información y, por lo tanto, la encargada de otorgar acceso para mostrar, modificar o solicitar información. Así, esta técnica busca determinar en qué clase se debe cumplir cierto requerimiento.

Sin embargo, algunas veces los requerimientos llevan varias tareas incluidas para llevarse a cabo. En este caso, los requerimientos se deben descomponer en requerimientos más pequeños que puedan ser llevados a cabo individualmente por las clases involucradas. Esto se conoce como la técnica de descomposición de los requerimientos. Por ejemplo, al pedir el perímetro de una figura, necesitábamos que la clase Lado determinara la distancia de cada arista de la figura y luego, con esta información, la clase Figura podía determinar su perímetro para luego retornarlo cada vez que lo necesitara la clase CajonFiguras. Así, la tarea de retornar el perímetro de una figura de la lista de figuras tenía que llevarse a cabo en tres procesos: 1) cálculo de la longitud de los lados en la clase Lado, 2) suma de las longitudes de los lados en la clase Figura, y 3) retorno del valor del perímetro en la clase CajonFiguras.

Adicional a esto, hay que identificar que estas tareas llevan ciertos requisitos que se deben cumplir en su asignación y/o modificación. Por ejemplo, si estoy asignando una edad en la clase Persona debo asegurarme de que sea un número entero positivo. A esta tarea se le denomina la asignación de responsabilidades.

ACTIVIDAD

1. Después de entender el enunciado del problema, se sugiere desarrollar un esquema del diseño del programa. Se puede hacer en un archivo de Word, señalando cada uno de los pasos en la solución del diseño del programa:
   a) Identificar mínimo cuatro requerimientos funcionales de la aplicación para la veterinaria. Se sugiere hacer más de cuatro, pero luego de terminar el resto de la actividad.
   b) Identificar las entidades del problema (las clases). Se sugiere que sea lo más general posible (4 clases a lo sumo).
   c) Identificar las características de las entidades (los atributos).
   d) Describir algunas relaciones entre las entidades.
   e) Identificar con la técnica de descomposición de requerimientos cuáles requerimientos necesitan ser descompuestos para poder cumplirse. Luego de esto, se debe asignar cada uno de los requerimientos (ya descompuestos) a las distintas entidades del problema (las clases).
   f) Finalmente, se deben identificar los requisitos para cada uno de los atributos de las clases con el fin de poder hacer la asignación de responsabilidades a cada método que corresponda.