# Práctico 3

## Paradigma Orientado a Objetos - Python

1. Modelar e implementar el objeto Perro, determinar sus atributos y métodos útiles. Realizar un programita que permita cargar 6 perros y las edades de cada uno, y determinar cual es el mayor de ellos, y la edad promedio de todos ellos.

2. Modelar e implementar la clase Vector de 2 componentes, y la clase Vector de 3 componentes. ¿Qué atributos deberían tener? ¿Qué métodos deberían tener?

3. Modelar e implementar el objeto Vehiculo. Tanto sus atributos como sus métodos. Realizar un programita que luego permita cargar los autos favoritos y los autos más odiados de su grupo de amigos.

4. Crear un Objeto Temperatura. Deberá tener como métodos, la posibilidad de conversión a diferentes temperaturas, además deberá permitir definir cuál será la temperatura por defecto que utilizará. Grados Celcius, Farenheit, Kelvins, etc.

5. Crear el objeto Empleado, deberá determinar cuáles serían sus atributos y métodos, considerando que se necesitará calcular su salario, y por lo tanto, información como horas trabajadas y tarifa por hora, serán importantes.

6. Modelar el objeto Celular. Luego, realizar un programa que permita ingresar la cantidad de dinero que el cliente dispone y determinar cuales modelos de la tienda podría comprarse, suponiendo que la tienda tiene 6 modelos a la venta.

7. Modelar e implementar la clase Estudiante. Qué atributos y métodos considera debería tener?

8. Implemente un programa que permita cargar 6 estudiantes, donde cada uno tiene las notas de sus examenes finales. Pueden ser diferentes entre un estudiante y otro, ya que cada uno ha rendido o promocionado diferente cantidad de materias. Modifique la clase Estudiante si fuese necesario. Realice un programita que le permita cargar 3 estudiantes y determine cual es el alumno que será el abanderado, si lo hubiese.

9. Modelar e implementar una clase para un personaje de un juego de aventura. Qué atributos debería tener? Qué métodos podría tener?

10. Crear una clase Texto que contenga métodos que le permitan invertir una frase, obtener la primer palabra, obtener la i-ésima palabra, chequear si es palíndrome o no. Contar la cantidad de una determinada letra pasada como argumento. *(No utilizar comandos chetos como reverse, count, split de los str ni de las listas. Desarrollar los métodos en base a razonamiento y las estructuras básicas, fundamentales.* Crear al menos un método propio que usted encuentre interesante para un texto. ¿Qué podría ser divertido o útil de realizar con un texto?

11. Crear una clase Carrito de compras, que permita agregar items, quitar items, determinar el costo total, y determinar si tiene envío gratis o no en base al monto total.

12. Modelar los objetos *TarjetaCredito* y *Cliente*. Además de determinar los atributos y posibles métodos de cada uno de esos objetos, debe prestar atención a cómo podrían estar relacionados entre si. Podría suponer que un cliente tiene solo una tarjeta de crédito por ejemplo, dentro de mi tienda, o que le permitiré tener todas las que necesite y de las cuales disponga. ¿Cómo deberá modelar dicha situación?

13. Implemente una clase llamada *MensajeCripto* con métodos *encriptar* y *desencriptar*. Deberá utilizar un esquema para encriptación básico de *Cifrado por Sustitución, como por ejemplo el Cifrado César*, de asignar caracteres de un conjunto específico al alfabeto español, para poder re escribir dicho texto. Y luego poder descifrarlo.
Deberá desarrollar los algoritmos Usted, de modo artesanal, sin caer en la tentación de utilizar librerias para criptografía ya existentes.
Sería ideal, que antes de investigar bien y mirar el material compartido por la cátedra, intente pensar por sí mismo en qué consistiria el encriptamiento o codificación de un mensaje, y su correspondiente desencriptado o decodificación del mismo.

14. ¿Cómo haría para modelar el objeto etiqueta *HTML*? Investigue, cuál es el conjunto de etiquetas de HTML puro y básico. Piense entonces, como implementar un objeto EtiquetaHTML que permita construir etiquetas HTML, que sean válidas y que tengan el contenido que va dentro de ellas. Simplifique tanto la cantidad de etiquetas de contemplará como los valores y opciones que pueden tomar dentro de ellas.

15. Crear la clase *Binario*. Esta clase debe permitir trabajar con números binarios, sumarlos, restarlos, etc., además de permitir la conversión hacia y desde números en base 10.

## Herencia

*a*) **Lavadero.**
En base a la clase Vehiculo implementada en el práctico anterior, revise su diseño, y determine si ahora, tuviese que utilizarla como clase Padre de los siguientes vehículos disponibles en el lavadero: auto particular, camioneta, moto, omnibus.
¿Cómo debería reformarla? Diseñe las clases para cada tipo de vehículo, y aplique herencia, para finalmente poder representar todos los vehículos que el lavadero acepta.
¿Cómo quedaría el Diagrama de Clases?

*b*) **Shelter de Mascotas**.
Se desea desarrollar un sistema para un Shelter de mascotas *(Shelter: refugio temporal para mascotas que se espera sean adoptadas)*. Los animales que este shelter cuida temporalmente y trata de ubicar en adopción son: perros, gatos, conejos, canarios, tortugas y peces.
Implementar una clase *Animal* que tendrá los siguientes atributos genéricos para cualquiera de los animales que interesarán en este programa: raza, nombre, color, edad, estado de salud, género.
El estado de salud, deberá ser implementado como un atributo que solo puede tomar uno de los valores posibles de una lista de opciones, cuyos valores serán diferentes según el tipo de animal, pues peces no tienen los mismos problemas que los perros por ejemplo.

Para animales con patas, el estado de salud podría contar con los siguientes valores posibles: sano, enfermo crónico, cojo, débil.
Para los peces, podría ser: sano, aleta_herida.
Para los pájaros, podría ser: sano, plumaje *(fuerte, débil)*, ala_herida.
A partir del siguiente Diagrama de Clases incompleto, que solo muestra la relación entre las clases, termine de diseñar cada clase, para poder así, completar el Diagrama de Clases como corresponde.

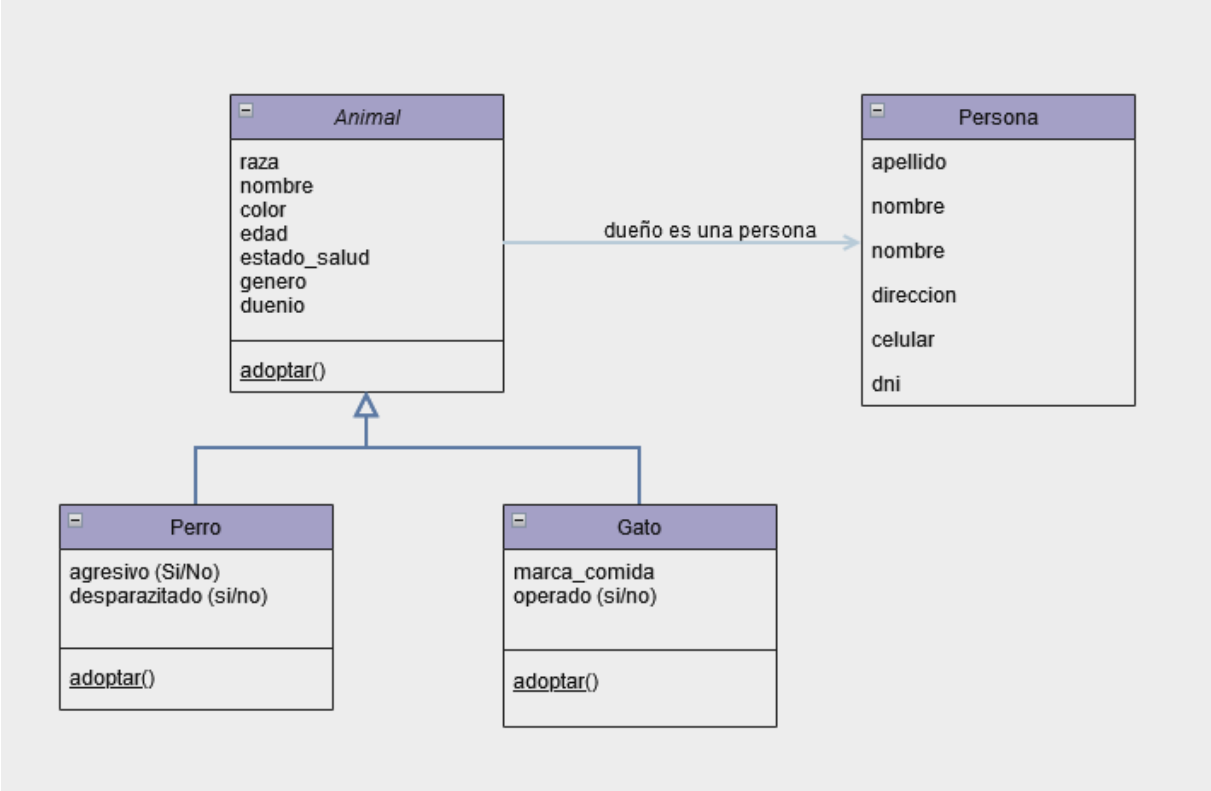


Figura 1: Diagrama de Clases *(incompleto)*.

Analice bien los atributos que podrían hacer falta. Diseñe para los peces y las aves, sus atributos. El método *adoptar*() no es implementado en la clase Padre *Animal*, sino en cada hijo. Determine si le hará falta un atributo *adoptado* (si/no) en el Animal.
Luego, cuando ya haya diseñado e implementado las clases, el siguiente paso será, crear un programa principal, para la administración de los animales en el Shelter. Suponga que al Shelter han llegado 2 conejos nuevos. Y que ya había 3 perros, un gato, 5 peces. Simule ahora como sería el proceso de adopción para 1 conejo y el gato, además de un perro, siempre y cuando no sea agresivo y pueda estar con niños.

*c*) **Empresa**.
Modele el sistema de usuarios de la empresa, teniendo en cuenta que:
Hay una clase llamada *Persona* que modela lo básico de una persona cualquiera, con atributos como dirección, edad, dni, celular, email, fecha de nacimiento. Esa clase deberá ser usada, para desarrollar las clases que representan: clientes, empleados, y

finalmente técnicos.
¿Qué atributos deberá tener si o si, un empleado? *(considere número de legajo, sección de la empresa en la que trabaja, disponible para horarios extras, antiguedad)*
¿Qué atributos podrían tener los clientes? *(considere por ejemplo, es cliente frecuente o no, ultima fecha de visita)*
Respecto a los empleados, debe modelar el hecho de que, hay diferentes niveles de acceso al sistema de la empresa: administrador *(tiene acceso a todo, puede agregar, modificar o eliminar empleados y clientes)*, gerente *(puede solo modificar datos de empleados existentes y ver todos los empleados que hay)*, vendedor *(puede solamente agregar clientes nuevos)*, técnico *(solo puede ver sus propios datos)*
Diseñe además, el Diagrama de Clases, una vez que tenga diseñada cada clase.
Ahora, implemente un programa principal, que utilizando todo dicho sistema de clases, genere un menú con diferentes opciones según qué tipo de usuario se ha logueado. El usuario, verá en primera instancia, un login, y al entrar con su usuario y contraseña, tendrá el menú de operaciones que puede realizar, acorde al rol que tiene dentro de la empresa. Cargar unos 5 o 6 usuarios para poder probar el sistema funcionando.

Extienda el programa, para que pueda dejar cargados los datos de todos los usuarios en un archivo de texto, llamado empresa.txt, cada fila del archivo de texto, pertenecerá a los datos que una persona. Separe los datos por coma, y organice la información de modo tal, que la primer columna del archivo, sea para el apellido, la segunda para el nombre, etc, para todas las personas guardadas en el archivo.
Ahora, adapte el programa principal de modo tal que, al agregar una persona, sea agregada al archivo. Y que al eliminar una persona, sea buscada dentro del archivo, y de algún modo, eliminada del mismo *(requerirá ingenio)*
Como siempre, no utilizar librerías sofisticadas, sino, simplemente lo visto en clase.

## Polimorfismo y Sobrecarga de Operadores

*d*) Utilizando las clases implementadas en el práctico anterior para vectores de 2 y 3 componentes, extiendalas ahora, para que los operadores aritméticos convenientes, sirvan con objetos del tipo vector.

*e*) Implementar la clase Persona, que entre los atributos clásicos de una persona, deberá tener el atributo de edad. Definir la operación mayor y menor que, entre dos personas, claramente hará referencia a la edad. *(ie., deberá sobrecargar los operadores relaciones $<, <=, >, >=$ para poder aplicarlos entre personas).*
Luego, crear un programa principal, que haciendo uso del tipo de datos Persona, permita ingresar al usuario, los datos de pares de invitados a la fiesta, entrarán de a dos personas a la fiesta, teniendo en cuenta que uno de ellos debe ser mayor que el otro, por al menos 10 años de diferencia y representará el tutor. El programa deberá indicar si esas dos personas, según la edad de cada uno, están habilitados para ingresar a la fiesta o no.

*f*) Implementar la clase *Complejo*, para representar un número complejo. Realizar además, sobrecarga de operadores, para poder sumar, restar, etc números complejos.

*g*) Dada la clase para representar números binarios, en el práctico anterior, extienda su funcionalidad, agregando sobrecarga de operadores aritméticos, para poder sumar, restar, etc dos números binarios, con los operadores aritméticos tradicionales.