Segundo Parcial de Programación 1

04 / 11 / 2022

Tener en cuenta que:

- La duración máxima del parcial es de 2 horas 30 minutos (150 minutos).
- Se debe realizar en computadora, para este fin se deberá seguir los siguientes pasos:
    - Crear una carpeta dentro de su computadora de nombre **segundoparcial2022**.
    - Dentro de esta carpeta se deben crear **todos los módulos** necesarios para la resolución del problema planteado.
    - Culminado las horas del parcial se deberá entregar a través del Moodle en la tarea **"Entrega Segundo Parcial"**. Se espera que se suba un Zip de la carpeta "**segundoparcial2022**" creada en el paso 1. Puede usar de nombre del zip **segundoparcial2022.zip**.
- Tenga en cuenta que no se corrige código con errores del intérprete de Python.
- La prueba es individual y se puede usar material.
- Se contestan dudas solo en la primera hora del parcial.
- El puntaje se encuentra distribuido uniformemente en todos los ejercicios.

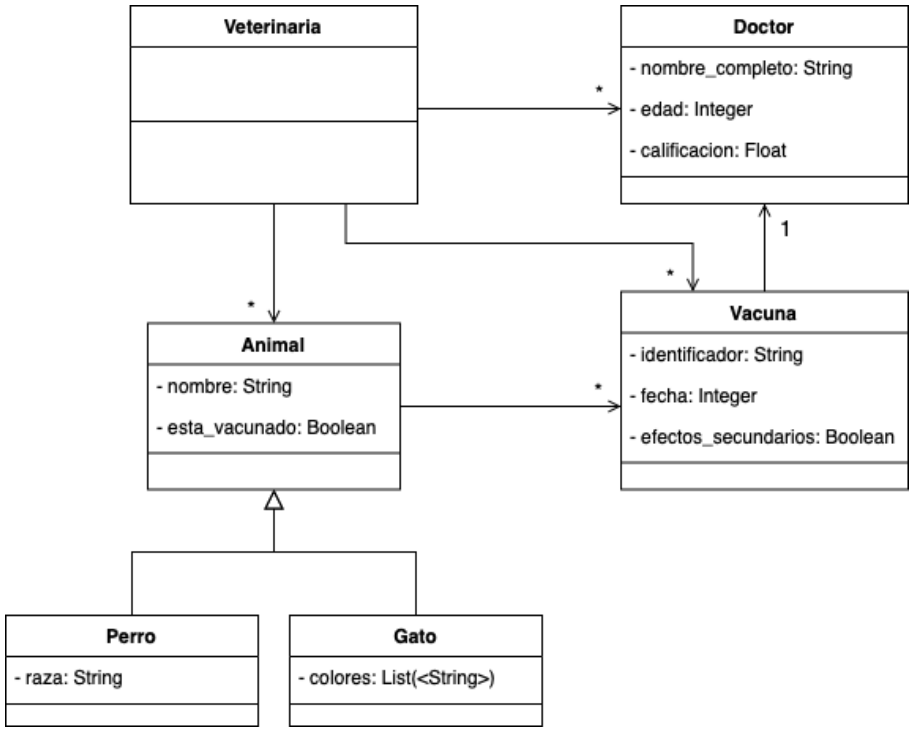
EJERCICIO: Sistema de Gestión de una Veterinaria

La veterinaria Pet Shop UY mantiene registros actualizados de sus empleados (doctores especializados en medicina de mascotas), animales atendidos (perros o gatos) y las vacunas efectuadas a los animales por algún doctor.

En este sistema personalizado, cada doctor posee un nombre completo (nombre y apellido), edad (25 a 80) y calificación (0 a 5) asociada. Luego, cada animal posee un nombre y raza (si es perro) o colores (si es gato) junto a una condición de estar suficientemente vacunados. Por último, cada vacuna posee un código identificador hexadecimal y una fecha o año de aplicación junto a una condición de generar efectos secundarios.

Se desea aquí generar un sistema que facilite la gestión de doctores, animales y vacunas, al igual que permita realizar ciertas funciones elementales como registrar animales con vacunas, validar vacunas de mascotas y sus efectos secundarios, calificar a los doctores por su labor y reemplazar doctores vinculados a determinadas mascotas.

Se propone el siguiente diagrama de clases parcial que modela la realidad a implementar:



Con esta realidad en mente, se debe implementar la clase Veterinaria que tenga las siguientes operaciones:

- **def registrar_animal_con_vacuna (self, nombre_animal, raza, colores, identificador, fecha, nombre_doctor, edad_doctor):** Se debe registrar el animal especificado por nombre y raza (Perro) o colores (Gato) en conjunto con su vacuna (definida por identificador y fecha) y doctor responsable con nombre y edad.
    - Si la raza es indicada, el animal a especificar debe ser un Perro. En caso contrario, el animal será un Gato con los colores indicados.
    - Si el animal ya se encuentra registrado en el sistema, actualizar sus vacunas. En caso contrario, registrar completamente al animal.
    - Si el doctor ya se encuentra registrado en el sistema, asociarlo a la vacuna. En caso contrario, registrar completamente al doctor.
    - **Exceptions:** Lanzar la excepción *InformacionInvalida* si algún argumento obligatorio es vacío o incorrecto. Lanzar la excepción *EntidadYaExiste* si el animal ya se encuentra vacunado con la vacuna especificada.

- **def validar_vacunas_y_efectos_secundarios (self, raza, color, fecha_inicial, fecha_final):** Corroborar que todas las vacunas no tengan efectos secundarios (modificando atributo *efectos_secundarios*) y que los animales indicados (todos los perros de raza y gatos de color especificados) posean una cantidad de vacunas suficientes sin efectos secundarios (modificando atributo *esta_vacunado*).
    - Una **vacuna** tienen efectos secundarios si su fecha se encuentra entre *fecha_inicial* y *fecha_final*, con extremos incluidos. En caso contrario, la vacuna no tiene efectos secundarios.
    - Un **perro** está vacunado si posee una cantidad mayor o igual a 3 vacunas sin efectos secundarios. Por otro lado, un **gato** está vacunado si posee una cantidad mayor o igual a 2 vacunas sin efectos secundarios.
    - **Exceptions:** Lanzar la excepción *InformacionInvalida* si algún argumento obligatorio es vacío. Lanzar la excepción *EntidadNoExiste* si no existe ningún perro de raza ni gato de color indicados que se encuentre vacunado.

- **def calificar_doctores_de_perros (self, nombre_perro, nombre_doctor, calificacion):** Modificar la calificación del doctor especificado por su nombre que se encuentra asociado al perro de nombre indicado.
    - **Exceptions:** Lanzar la excepción *InformacionInvalida* si algún argumento obligatorio es vacío o incorrecto. Lanzar la excepción *EntidadNoExiste* si el perro indicado no se encuentra asociado al doctor especificado.

- **def cambiar_doctor_de_animal (self, doctor_anterior, nombre_animal, doctor_nuevo):** Para el animal con *nombre_animal*, reemplazar el doctor indicado por *doctor_anterior* con el doctor especificado por *doctor_nuevo*.
    - Tanto *doctor_anterior* como *doctor_nuevo* son instancias de Doctor.
    - **Exceptions:** Lanzar la excepción *InformacionInvalida* si algún argumento obligatorio es vacío o si ambos doctores son iguales. Lanzar la excepción *EntidadNoExiste* si el perro indicado no se encuentra asociado al doctor anterior especificado.

Consideraciones:

- Todas las clases sin ser las excepciones y la clase Veterinaria se deben colocar en el paquete **entities**. La clase Veterinaria se debe colocar en la **raíz** del proyecto. Las excepciones se deben colocar en el paquete **exceptions**.
- Se debe preservar el encapsulamiento de todas las entidades, para las cuales se deben usar getters (todas las propiedades) y setters (si son realmente necesarios).
- Se debe crear obligatoriamente un módulo por clase.
- Se debe crear el módulo **gestion_veterinaria.py** una operación “main” que pruebe el correcto funcionamiento de las operaciones implementadas. Se esperan al menos dos pruebas de cada operación.