Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2023-1

30 de Marzo de 2023
Actividad

Actividad 1

Programación Orientada a Objetos II

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/A1/`
- **Hora del *push*:** 16:40

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Introducción

Luego de defender exitosamente el reino, para enmendar el daño hecho a sus tortugas, decides ayudar al mundo de los veterinarios a subirse al mundo de la tecnología. Es por esto, que te decides a preparar e implementar la modelación de distintas entidades de animales.

Archivos

Archivos de código

En el directorio de la actividad encontrarás los siguientes archivos con código:

- Modificar `clases.py`: Aquí encontrarás la definición básica de las clases que debes implementar y completas.
- No modificar `test.py`: Este archivo contiene los tests que podrás utilizar pasa ir viendo si lo desarrollado hasta el momento cumple con lo pedido por los veterniarios.

Modelo de datos

Para modelar correctamente el problema, desde la veterinaria te entregan las siguientes especificaciones escritas.

Se necesita un programa que, por el momento, pueda representar a los distintos animales que atendemos. En nuestro consultorio por ahora solo atendemos animales terrestres y acuáticos, pero podríamos eventualmente atender más. Algo que tienen en común todos los animales, es que todos tienen un peso, un identificador único y además cuando llegan tienen un nivel de energía en 100. Además de eso, todos los animales pueden desplazarse, pero cada uno lo hace a su propia manera.

Respecto a los animales de tipo Terrestre, estos además de ser animales como todos, poseen una cantidad de patas, y tienen un gasto energético por desplazamiento dado por un número aleatorio entre 1 y 10 multiplicado por su peso. Esto quiere decir que, cada vez que los animales terrestres se desplazan, su energía disminuye en un número aleatorio entre 1 y 10.

Respecto a los animales de tipo acuático, estos no poseen características extras, pero si tienen un gasto energético por desplazamiento. El gasto energético por desplazamiento está dado por un numero aleatorio entre 1 y 3 multiplicado por el peso. Esto quiere decir que, cada vez que los animales acuáticos se desplazan, su energía disminuye en dicha cantidad.

Actualmente, entre los animales puramente terrestres, atendemos solo a los perros. Todos los perros tienen 4 patas, y además de las características propias de los animales terrestres, poseen una raza y tienen la capacidad de ladrar.

Por otro lado, de los animales acuáticos solo atendemos peces. Estos además de poseer las características de los animales acuáticos, poseen un color y pueden nadar.

Finalmente, tenemos una especie media extraña que lo tenemos categorizado como animal acuático y terrestre a la vez, que es el ornitorrinco. Este posee las características tanto de animales acuáticos como terrestres, y su forma de desplazar es distinta a todos los demás. El gasto energético asociado al desplazamiento, es el promedio (redondeado a número entero) resultante entre el desplazamiento del terrestre y el desplazamiento del acuático.

Para facilitar tu trabajo, te entregaron el siguiente diagrama de clases.

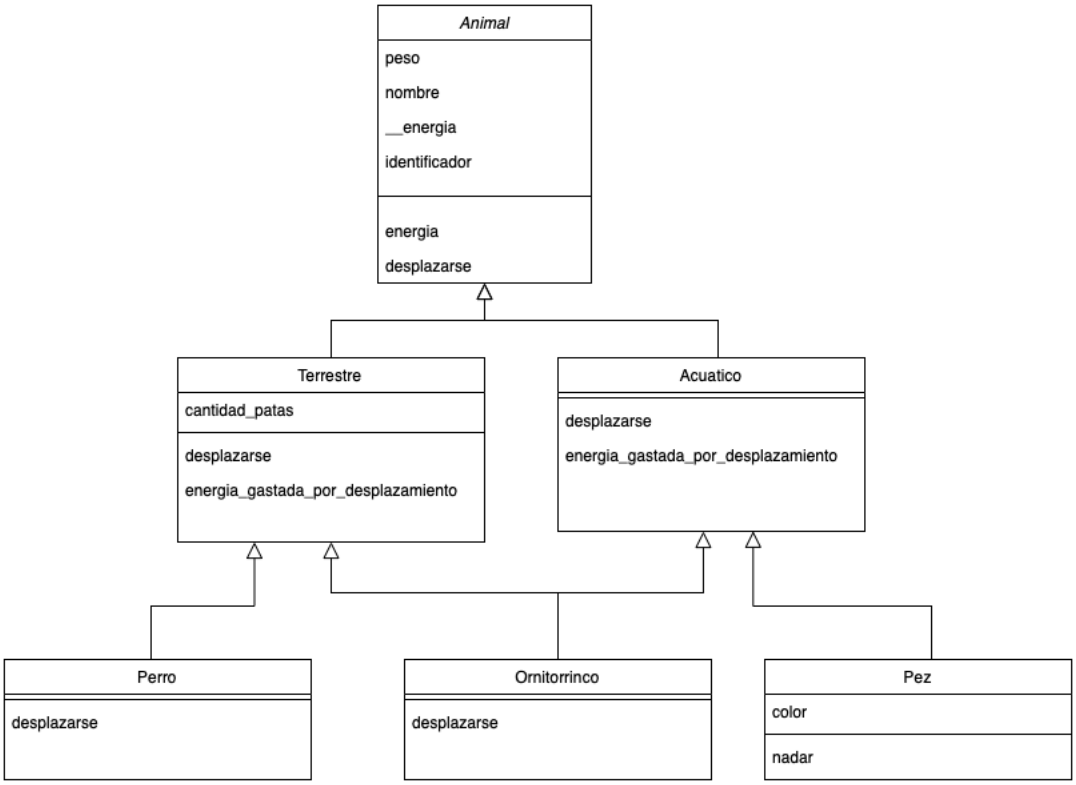

Figura 1: Diagrama de clases

Llevando modelo al código

- Modificar `class Animal`:
  Clase abstracta que representa un animal. Posee la variable de clase `identificador`, y además incluye los siguientes métodos:

  - Modificar `def __init__(self, peso, nombre, *args, **kwargs) -> None`:
    Este es el inicializador de la clase, y debe asignar los siguientes atributos:

`self.peso`	Un `int` que representa el peso del animal.
`self.nombre`	Un `str` que representa el nombre de la mascota.
`self.__energia`	Un `int` que representa la energía de la mascota. Es una atributo privado que tiene un valor de 100 por defecto, y cuyo valor no puede bajar de 0.
`self.identificador`	Un `int` que sirve para identificar al animal. Para setearlo, debe fijarlo como el valor actual de la variable de clase `Animal.identificador`, y luego se le debe sumar 1 a la misma variable. De esta forma, se logra que todos los animales tengan un identificador diferente.

  - Modificar `def desplazarse(self) -> None`:
    Método abstracto, que obliga a las clases hijas a que este método sea implementado.

  - Modificar `def energia(self) -> int`:
    Property encargada de manejar la energía. El getter debe retornar el valor del atributo privado `self.energia`, y el setter debe encargarse de que el atributo privado no pueda tener un valor menor a 0.

- Modificar `class Terrestre`:
  Clase abstracta utilizada para representar animales terrestres. Hereda de animal, y posee los siguientes métodos:

  - Modificar `def __init__(self, cantidad_patas, mensaje_desplazamiento, *args, **kwargs) -> No`
    Este es el inicializador de la clase. Además de llamar al método de la clase padre, debe asignar los siguientes atributo:

`self.cantidad_patas`	Un `int` que representa la cantidad de patas que tiene.

  - Modificar `def energia_gastada_por_desplazamiento(self) -> int`:
    Método encargada de calcular la energía necesaria para un desplazamiento. Debe retornar la energía requerida dada por la fórmula `self.peso * 5`

  - Modificar `def desplazarse(self) -> str`:
    Método encargado de simular el desplazamiento de un animal. Al ser llamado, debe restar `energia_gastada_por_desplazamiento` a la energía actual del animal y retorna el texto `"caminando..."`

- Modificar `class Acuatico`:
  Clase abstracta utilizada para representar animales acuáticos. Hereda de animal, y posee los siguientes métodos:

  - Modificar `def energia_gastada_por_desplazamiento(self) -> int`:
    Método encargada de calcular la energía necesaria para un desplazamiento. Debe retornar la energía requerida dada por la fórmula `self.peso * 2`

  - Modificar `def desplazarse(self) -> str`:
    Método encargado de simular el desplazamiento de un animal. Al ser llamado, debe restar `energia_gastada_por_desplazamiento` a la energía actual del animal y retorna el texto `"nadando..."`

- Modificar `class Perro`:
  Clase que representa los perros. Debe heredar de la clase `Terrestre`:

  - Modificar `def __init__(self, raza, *args, **kwargs) -> None`:
    Inicializador de la clase. Además de llamar al método de la clase padre, debe setear el atributo la raza.

  - Modificar `def ladrar(self) -> str`:
    Método que simula un ladrido. Debe retornar el texto `"guau guau"`.

- Modificar `class Pez`:
  Clase que representa los peces. Debe heredar de la clase `Acuatico`:

  - Modificar `def __init__(self, color, *args, **kwargs) -> None`:
    Inicializador de la clase. Además de llamar al método de la clase padre, debe setear el atributo color.

  - Modificar `def nadar(self) -> str`:
    Método que simula un nado. Debe retornar el texto `"moviendo aleta"`.

- Modificar `class Ornitorrinco`:
  Clase que representa los ornitorrincos. Debe heredar de las clases `Acuatico` y `Terrestre`. Posee el siguiente método
  - Modificar `def desplazarse(self) -> str`:
    Debe desplazarse como animal acuático y animal terrestre. El gasto de energía debe ser el promedio de ambos animales (redondeado a entero). Debe retornar la concatenación de los dos métodos de desplazarse de las clases padres.

Notas

- Recuerda que la ubicación de tu entrega es en tu **repositorio personal**. Verifica que no estés trabajando en el **Syllabus**.
- Se recomienda completar la actividad en el orden del enunciado.
- Si aparece un error inesperado, ¡léelo! Intenta interpretarlo.

Objetivos de la actividad

- Implementar clases siguiendo especificiones técnicas.
- Hacer uso de clases y métodos abstractos para una correcta modelación.
- Manejar concepto de herencia y multiherencia correctamente.
- Hacer correcto uso de *properties*.
- Utilizar métodos de clases padres correctamente.
- Resolver bien el problema del diamante.