PROGRAMACIÒN ORIENTADA A OBJETOS

TALLER No. 5 PYTHON

PROGRAMACIÓN ORIENTADA A OBJETOS

Profesor: Jaime Alberto Guzmán Luna

Contenido del taller:

1) Herencia
2) Polimorfismo

1. EJERCICIO

```
1  class SerVivo:
2      _totalCreados = 0
3
4      def___init__(self, nombre, edad):
5          self._nombre = nombre
6          self._edad = edad
7          SerVivo._totalCreados += 1
8
9      def setNombre(self, nombre):
10         self._nombre = nombre
11
12     def getNombre(self):
13         return self._nombre
14
15     def setEdad(self, edad):
16         self._edad = edad
17
18     def getEdad(self):
19         return self._edad
20
21     @classmethod
22     def getTotalCreados(cls):
23         return cls._totalCreados
```

```
1  class Animal(SerVivo):
2      _totalCreados = 0
3
4      def___init__(self, nombre, edad, raza):
5          self._nombre = nombre
6          self._edad = edad
7          self._raza = raza
8          Animal._totalCreados += 1
9
10     def setRaza(self, raza):
11         self._raza = raza
12
13     def getRaza(self):
```

```
14          return self._raza
15
16      def setNombre(self, nombre):
17          self._nombre = nombre
18
19      def getNombre(self):
20          return self._nombre
21      def caminar():
22          pass
23
24      def correr():
25          pass
26
27      def ruido():
28          pass
29
30      @classmethod
31      def getTotalCreados(cls):
32          return cls._totalCreados
```

```
1   class Perro(Animal):
2       _totalCreados = 0
3
4       def__init_(self, nombre, edad, raza, pelaje):
5           super._init_(nombre, edad, raza)
6           self._pelaje = pelaje
7           Perro._totalCreados += 1
8
9       def setPelaje(self, pelaje):
10          self._pelaje = pelaje
11
12      def getPelaje(self):
13          return self._pelaje
14
15      def caminar():
16          return "camina muchos pasos"
17
18      def correr():
19          return "corre al punto"
20
21      def ruido():
22          print("guau")
23
24      @classmethod
25      def getTotalCreados(cls):
26          return cls._totalCreados
```

```
1   class Gato(Animal):
2       _totalCreados = 0
3
4       def__init_(self, nombre, edad, raza, pelaje):
5           super._init_(nombre, edad, raza)
6           self._pelaje = pelaje
7           Gato._totalCreados += 1
```

```
8
9       def setPelaje(self, pelaje):
10          self._pelaje = pelaje
11
12      def getPelaje(self):
13          return self._pelaje
14
15      def caminar():
16          return "se mueve lentamente"
17
18      def correr():
19          return "salta al punto"
20
21      def ruido():
22          print("miau")
23
24      @classmethod
25      def getTotalCreados(cls):
26          return cls._totalCreados
```

```
1   class Pajaro(Animal):
2       _totalCreados = 0
3
4       def__init_(self, nombre, edad, raza, colorAlas):
5           super._init_(nombre, edad, raza)
6           self._colorAlas = colorAlas
7           Pajaro._totalCreados += 1
8
9       def setColorAlas(self, colorAlas):
10          self._colorAlas = colorAlas
11
12      def getColorAlas(self):
13          return self._colorAlas
14
15      def caminar():
16          return "se mueve poco"
17
18      def correr():
19          return "no corre"
20
21      def volar():
22          return "vuela al punto"
23
24      @classmethod
25      def getTotalCreados(cls):
26          return cls._totalCreados
```

```
1   class Persona(SerVivo):
2       _totalCreados = 0
3
4       def__init_(self, nombre, edad):
5           super._init_(nombre, edad)
6           self._animalAcargo = None
7           Persona._totalCreados += 1
```

```
8
9       def aduenarAnimal(self, x):
10          self._animalAcargo = x
11          x.ruido()
12
13      @classmethod
14      def getTotalCreados(cls):
15          return cls._totalCreados
```

Para pensar…

1. Si deseo modificar el mensaje del método **ruido** al crear un objeto **Pajaro** sin alterar la clase **Animal** ¿Que debo agregarle al código? (Por ejemplo, al llamar el método **ruido** imprima **cantar y silbar**)
2. Si se crea una nueva clase Pez, y no se define nuevos métodos, constructor y atributos, ¿qué constructor tendrá esta clase, que argumentos recibe? ¿qué otros métodos y atributos tendrán estos mismos?
3. ¿Qué ocurre con el atributo nombre y edad de la clase SerVivo, al momento definirse en la Clase Animal? ¿Cómo cambiaria el código del constructor para que estos atributos sean el mismo?
4. En la clase Animal se sobrescribieron los métodos setNombre y getNombre,¿Como modificaría estos métodos para que su funcionamiento no oculte algún valor de la clase padre?¿podría plantearse esta solución usando super()?
5. El atributo totalCreados de la clase SerVivo ¿es heredado por las demás clases? ¿En este caso ocurre ocultación de los atributos al definirlo de nuevo en las clases hijas?
6. ¿Los métodos getTotalCreados sobrescriben al metodo de su padre?¿?
7. ¿Qué métodos hereda una clase que hereda de la clase Persona?
8. ¿Qué tipo de objetos podrían pasársele al método aduenarAnimal de la clase Persona?¿Se le puede pasar un objeto serVivo?
9. ¿Como podría obtener la cantidad de seres vivos que se creen (Objetos creados)?
10. Si se define el siguiente método en la clase Perro:

    def getRaza(self, tipo):
        return self._raza + " " + tipo

    ¿Qué ocurre con el método getRaza de la clase Animal? ¿Este método se sobreescribe o se sobrecarga?

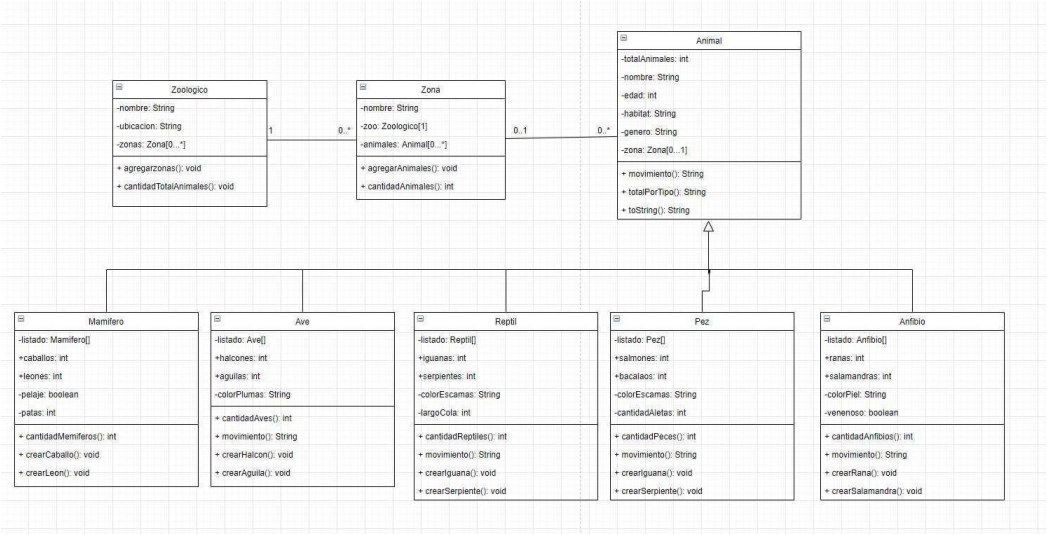
2. EJERCICIO PRACTICO GITHUB

Enlace entrega: https://classroom.github.com/a/0-sdksz3

Problema: Mi primer Zoológico

Queremos crear un software que lleve el control y modelado de mi zoológico, para esto se le dieron ciertas indicaciones:

PROGRAMACIÒN ORIENTADA A OBJETOS



- Se le brindo el anterior diagrama de clases, guíese para crear las clases, atributos y métodos que se le están pidiendo.
- Las clases Zoologico y Zona créelas en un paquete llamado gestion y las clases Animal y sus subclases añádalas al paquete zooAnimales.
- Para cada caso cree un constructor con los parámetros de los atributos proporcionados en cada clase en el orden que aparecen los atributos
- Para la clase Zoológico el método cantidadTotalAnimales, retornará la cantidad de animales total de todas las zonas que tengan relación con el Zoológico, agregarZonas será el método encargado de agregar nuevas zonas al zoologico.
- Para la clase Zona el método agregarAnimales añadira un nuevo animal al listado de animales y el método cantidadAnimales retornara la cantidad de animales en la zona.
- Para la clase Animal tenemos ciertas particularidades, queremos llevar el conteo del total de animales creados y que este valor podamos almacenarlo en el atributo totalAnimales, el método totalPorTipo, devolverá el siguiente formato con la cantidad de animales por cada subclase de animales:

  “Mamiferos: #
  Aves: #
  Reptiles: #
  Peces: #
  Anfibios: #“
  # es el numero de animales por cada subclase.

- El método toString de la clase Animal retornara el siguiente formato: “Mi nombre es #nombre, tengo una edad de #edad, habito en #habitat y mi genero es #genero, la zona en la que me ubico es #zona, en el #zoo” cambie respectivamente los valores contiguos con el #, tenga en cuenta que los valores de zoo y zona son los valores del nombre del respectivo objeto, en case de que el animal no tenga una zona solo imprima hasta el genero.

- Las subclases de Animal tendrán un atributo listado en donde se almacena cada objeto al ser creado, del tipo de la subclase.

PROGRAMACIÒN ORIENTADA A OBJETOS

Los siguientes métodos de creación retornaran el objeto creado:

- La clase Mamifero tendrá algunos comportamientos particulares, en el atributo caballos y leones se quiere llevar el conteo de veces que se usó el método crearCaballo y el método crearLeon y el metodo cantidadMamiferos retornara la cantidad de mamíferos creados en el sistema.
- Para la clase Mamifero el método crearCaballo, creará un Mamifero con los valores para pelaje = true, patas = 4 y hábitat = “pradera”, los demás valores los recibirá como parámetros.
- Para la clase Mamifero el método crearLeon, creará un Mamifero con los valores para pelaje = true, patas = 4 y hábitat = “selva”, los demás valores los recibirá como parámetros.
- La clase Ave tendrá algunos comportamientos particulares, en el atributo halcones y aguilas se quiere llevar el conteo de veces que se usó el método crearHalcon y el método crearAguila y el metodo cantidadAves retornara la cantidad de aves creadas en el sistema.
- Para la clase Ave el método crearHalcon, creará un Ave con los valores para colorPlumas = “cafe glorioso” y hábitat = “montanas”, los demás valores los recibirá como parámetros.
- Para la clase Ave el método crearAguila, creará un Ave con los valores para colorPlumas = “blanco y amarillo” y hábitat = “montanas”, los demás valores los recibirá como parámetros.
- La clase Reptil tendrá algunos comportamientos particulares, en el atributo iguanas y serpientes se quiere llevar el conteo de veces que se usó el método crearIguana y el método crearSerpiente y el metodo cantidadReptiles retornara la cantidad de reptiles creados en el sistema.
- Para la clase Reptil el método crearIguana, creará un Reptil con los valores para colorEscamas = “verde”, largoCola = 3 y hábitat = “humedal”, los demás valores los recibirá como parámetros.
- Para la clase Reptil el método crearSerpiente, creará un Reptil con los valores para colorEscamas = “blanco”, largoCola = 1 y hábitat = “jungla”, los demás valores los recibirá como parámetros.
- La clase Pez tendrá algunos comportamientos particulares, en el atributo salmones y bacalaos se quiere llevar el conteo de veces que se usó el método crearSalmon y el método crearBacalao y el metodo cantidadPeces retornara la cantidad de peces creados en el sistema.
- Para la clase Pez el método crearSalmon, creará un Pez con los valores para colorEscamas = “rojo”, cantidadAletas = 6 y hábitat = “oceano”, los demás valores los recibirá como parámetros.
- Para la clase Pez el método crearBacalao, creará un Pez con los valores para colorEscamas = “gris”, cantidadAletas = 6 y hábitat = “oceano”, los demás valores los recibirá como parámetros.
- La clase Anfibio tendrá algunos comportamientos particulares, en el atributo ranas y salamandras se quiere llevar el conteo de veces que se usó el método crearRana y el método

crearSalamandra y el metodo cantidadAnfibios retornara la cantidad de anfibios creados en el sistema.

- Para la clase Anfibio el método crearRana, creará un Anfibio con los valores para colorPiel = “rojo”, venenoso = true y hábitat = “selva”, los demás valores los recibirá como parámetros.
- Para la clase Anfibio el método crearSalamandra, creará un Anfibio con los valores para colorPiel = “negro y amarillo”, venenoso = false y hábitat = “selva”, los demás valores los recibirá como parámetros.

Nota1: note que, en las clases Zoológico, y Zona se creo un atributo adicional llamado zonas y zoo estos atributos no se ponen explícitamente dentro del diagrama normalmente, se sobre entiende de las relaciones que hay entre dos clases, esto quiere decir que si hay una relación entre X y Y, existirá un atributo para X de tipo Y, y en Y uno de tipo X, y este será un listado dependiendo de la cardinalidad (0..* o 1..*).
Nota2: los atributos que están definidos como private (-)ta debe crear sus métodos get y set.

Para pensar…

1. ¿Por qué es útil la herencia en Python?
2. ¿Ves mucha diferencia de la herencia respecto a Java?