Respuestas Punto 2- Taller 5 Python- Julián Bedoya Palacio.

Preguntas de análisis

1. Si deseo modificar el mensaje del método ruido al crear un objeto Pajaro sin alterar la clase Animal ¿Qué debo agregarle al código? (Por ejemplo, al llamar el método ruido imprima, cantar y silbar).

Se debería agregar las siguientes líneas en la clase pájaro:

```
def ruido(self):
    return "cantar y silbar"
```

En este caso se sobrescribe dicho método y al llamarlo desde el objeto creado de tipo Pájaro, se opaca la definición del padre.

2. Si se crea una nueva clase Pez, y no se definen nuevos métodos, constructor y atributos. ¿Qué constructor tendrá esta clase, qué argumentos recibe? ¿Qué otros métodos y atributos tendrán estos mismos?

Al ser una clase hija de Animal, se heredan todos los métodos de su padre (Animal) y hereda de Objet; en cuanto al primero, el __init__ que recibe como parámetros (nombre, edad, raza), se debe tomar en cuenta que para llamarlo se debe hacer por super().__init__(…), los métodos getters y setters de raza y nombre, los métodos caminar, correr y ruido y los atributos _totalCreados, nombre, edad, y raza; finalmente de objetc hereda el __str__().

3. ¿Qué ocurre con el atributo nombre y edad de la clase SerVivo, al momento de definirse en la Clase Animal? ¿Cómo cambiaría el código del constructor para que estos atributos sean el mismo?

Lo que ocurre es que se redefinen y al ser atributos de instancia solo se crea un espacio de memoria, por lo que si se modifican en cualquiera de las dos clases, el valor que toman ambos es el último que se le asigna.

Debido a lo mencionado anteriormente, no existe necesidad de cambiar el código, ya que son los mismos.

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()?

De hecho sí, para que no se oculte lo que sucede en los métodos del padre (SerVivo), es necesario agregar super(). en cada método (según corresponda) pasando el parámetro nombre en el caso del set y ningún parámetro en el caso del get

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?

Si es heredado por las demás clases y se puede acceder desde el nombre de la clase, en este caso SerVivo (lo más recomendado)

No ocurre ocultación debido a que los atributos de clase son propios de la clase, y al intentar modificarlo, solo cambiará para la clase que se referencie.

6. ¿Los métodos getTotalCreados sobrescriben al metodo de su padre?

Sí lo sobrescriben pero no lo ocultan como en Java.

7. ¿Qué métodos hereda una clase que hereda de la clase Persona?

Hereda el __init__, aduenarAnimal, y getTotalCreados, además los métodos que Animal hereda de Ser vivo que son: setNombre, getNombre, setEdad, getEdad, getTotalCreados y los métodos de object en específico el __str__

8. ¿Qué tipo de objetos podrían pasársele al método aduenarAnimal de la clase Persona?¿Se le puede pasar un objeto serVivo?

Se le podrían pasar objetos de tipo Animal, Gato, Pájaro, Perro y serVivo, este último es válido porque en Python no hay restricciones sobre los tipos de parámetros como sucede en Java

9. ¿Cómo podría obtener la cantidad de seres vivos que se creen (objetos creados)?

Haciendo uso de la siguiente instrucción:

SerVivo.getTotalCreados()

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?

Este método se sobrescribe por lo que las instancias de la clase hija conocerán el método de la propia clase, y no el que está en Animal; sin embargo, getRaza seguirá siendo el de Animal para otras subclases de esta última.