EJERCICIO 2 – TALLER 5 DE PYTHON

Luis Esteban Rincon Jaimes ---- C.C. 1090384822

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 debe sobrescribir el método ruido() en la clase Pajaro, así:

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

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?

Esta nueva clase "vacía", suponiendo que también hereda de la clase Animal, tendrá como inicializador el de su clase padre (clase Animal), por ende, si se crea un objeto de tipo Pez, se tendrán que pasar los parámetros que normalmente se le pasan al inicializador de la clase Animal, esto es debido a que esta nueva clase Pez lo estaría heredando (porque el __init__ también es otro método, entonces se puede heredar). Además, esta clase heredará también todos los atributos y métodos de la clase Animal (los que no sean privados).

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?

Los atributos nombre y edad de la clase serVivo son redefinidos en la clase Animal, cuando se crea un objeto de tipo Animal. Para cambiar el código del **inicializador** de manera que estos atributos sean el mismo, debemos redefinir el constructor de la clase Animal de la siguiente forma:

```
def __init__(self, nombre, edad, raza):
    super().__init__(nombre, edad)
    self._raza = raza
    Animal._totalcreados +=1
```

Con esto, se le envían los atributos nombre y edad a los atributos definidos en su clase padre (clase serVivo), lo que hace que estas dos clases tengan los mismos valores en sus atributos nombre y edad.

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()?

Sí, de la siguiente manera:

```
def setNombre(self, nombre):
    self._nombre = nombre
    super().setNombre(nombre)
def getNombre(self):
    return super().getNombre()
```

Se usan los métodos setNombre y getNombre de la clase padre (clase serVivo) porque el atributo nombre es privado.

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?

Este atributo es heredado, pero al redefinirse en sus clases hijas (Persona, Animal, Gato, Pajaro, Perro), se oculta el atributo de la clase padre. Esto causa que cada clase mantenga su propio contador independiente.

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

Sí, ya que el método getTotalCreados() está definido con el mismo nombre en cada clase hija de la clase serVivo (Persona, Animal, Gato, Pajaro, Perro), sobrescribiendo el método de la clase padre. Esto hace que cada subclase retorne solo su propio contador.

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

Cualquier clase que herede de la clase Persona obtendrá todos sus métodos, incluyendo: su método __init__(), aduenarAnimal(), getTotalCreados() y los métodos heredados de SerVivo: setNombre(), getNombre(), setEdad(), getEdad().

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

El método aduenarAnimal() puede recibir cualquier objeto que tenga definido el método ruido() en él, puesto que en la línea 13 intenta llamarlo explícitamente en el objeto que sea pasado como parámetro, entonces si dicho objeto no cuenta con este método se generará un error. Los objetos que cuentan con este método ruido() y pueden ser pasados como parámetro al método aduenarAnimal() son los objetos de la clase Animal y los de sus clases hijas (Perro, Gato y Pajaro).

No se le puede pasar un objeto serVivo porque este no cuenta con el método ruido(), por ende, si se pasa un objeto serVivo, cuando en la línea 13 se intente llamar a dicho método se generaría un error.

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

Para esto se debe dejar solamente un atributo de clase llamado "_totalCreados" en la clase que heredan todas las demás (clase serVivo), haciendo que este atributo sea único y lleve el contador de todos los objetos creados (incluyendo los de sus clases hijas y las que hereden de ellas).

Por lo anterior, para acceder a la cantidad de seres vivos que se creen se deben eliminar los demás atributos de clase llamados "_totalCreados" creados en las subclases de SerVivo (y en las subclases de ellas) y, además, cambiar la sobrescritura de los métodos getTotalCreados() en cada una de ellas, de la siguiente manera:

```
@classmethod
def getTotalCreados(cls):
    return super().getTotalCreados()
```

con esto, se puede imprimir la cantidad TOTAL de seres vivos creados así:

```
print(SerVivo.getTotalCreados())
```

o desde cualquier clase hija de la clase SerVivo y las que hereden de ellas así:

```
print(Animal.getTotalCreados())
print(Persona.getTotalCreados())
print(Gato.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?

Según lo anterior, el método getRaza() se sobrescribe, debido a que en Python no existe la sobrecarga de métodos, porque este lenguaje no tiene en cuenta (no diferencia) el tipo de dato que se le ingresan a los parámetros. Por lo anterior, si hay dos métodos que se llaman igual en una misma clase, esta clase se quedará solamente con la última definición de dicho método.

Con esto, concluimos que el nuevo método getRaza() en la clase Perro sobrescribe el método getRaza() heredado de la clase Animal, de la forma que nos indicaron arriba, pidiendo obligatoriamente un parámetro al ser invocado.