Taller 5 python ejercicio 2

Jhoneyker Delgado Urbina

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).**
   Si se desea modificar el mensaje del método ruido al crear un objeto Pajaro sin alterar la clase Animal se debe agregar a la clase Pajaro un método llamado ruido() de la siguiente manera:

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

   De está manera python oculta el método ruido() de la clase padre y utiliza el método ruido() de Pajaro e imprimirá lo que esta dentro del print().

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?**
   Siguiendo la lógica la clase Pez heredaría de la clase Animal y al no definirse nada nuevo entonces tendría el inicializador (__init__(self, nombre, edad, raza)) de la clase padre, además, tendría los métodos setRaza(), getRaza(), setNombre(), getNombre(), caminar(), correr(), ruido() y getTotalCreados, y si se tiene encuenta la privacidad de los atributos este no heredaría ningún atributo ni de clase ni de instancia, dejando esto a un lado y teniendo en cuenta que Python no conoce esto entonces herada los atributos _totalCreados, _nombre, _edad y _raza.

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?**
   En teoría los atributos no se deberían de heredar ya que "son privados" por lo que al definirse en la Clase Animal deberían ser distintos espacios de memoria y para que sean el mismo debería quitarse el _ de los atributos _nombre y _edad.

   En Python como no conoce que es la "privacidad" entonces hereda estos atributos y se ocultan para la clase al momento de redefinirse en la clase Animal.

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()?**
   Si se puede dar solución a esto utilizando super() de la siguiente forma:
   Para el método setNombre(self,nombre)**:**

```
super().setNombre(nombre)
```

Y para el método getNombre(self)**:**

```
return super().getNombre()
```

De esta manera no se ocultan los valores correspondientes de los métodos de la clase padre.

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?**
   Sí se tiene en cuenta que el atributo “es privado” no se heredaría y no ocurriría ocultamiento al definirlo de nuevo en las clases hijas, pero en la practica Python no conoce privacidad por lo que el atributo totalCreados es heredado por las clases hijas de este, pero ocurre ocultamiento de este atributo ya que se define de nuevo en las clases hijas por lo que Python toma el atributo totalCreados de la clase en la que se encuentre.

6. **¿Los métodos getTotalCreados sobrescriben al metodo de su padre?**
   Sí, los métodos getTotalCreados sobreescriben al método de su padre, ya que en Python, al declarar un método con el mismo nombre que un método de la clase padre se sobrescribirá en el contexto de la subclase, sin alterar al método de la clase padre.

7. **¿Qué métodos hereda una clase que hereda de la clase Persona?**
   Una clase que hereda de la clase Persona hereda, en primera instancia, el método inicializador __init__() definido en Persona, así como el método de instancia aduenarAnimal() y el método de clase getTotalCreados() (también de Persona). Además, hereda los métodos que Persona a su vez heredó de SerVivo, que son: setNombre(), getNombre(), setEdad() y getEdad().

8. **¿Qué tipo de objetos podrían pasársele al método aduenarAnimal de la clase Persona?¿Se le puede pasar un objeto serVivo?**
   A nivel teórico solo se le podría pasar objetos de la clase Animal y los que sean hijos de este como objetos de las clases Gato, Pajaro y Perro ya que estos se pueden comportar también como Animal. Pero no se le puede pasar un objeto serVivo ya que es un nivel más alto.

   Y a nivel practico se le puede pasar cualquier tipo de objetos pero mostraría error al momento de invocar el método ruido si el objeto que se le pase no lo tiene definido.

9. **¿Cómo podría obtener la cantidad de seres vivos que se creen (objetos creados)?**
   Para obtener la cantidad de seres vivos que se creen primero se debe agregar al inicializador de la clase Animal que es hija de SerVivo un:

   ```
   super.__init__(nombre,edad)
   ```

Para que así la clase SerVivo tenga un registro de todos los objetos que se creen tanto de Persona como de Animal y de esta manera haciendo llamado al método de clase getTotalCreados() de la clase SerVivo muestre la cantidad de objetos que se han creado.

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?

El método getRaza se sobreescribe para la clase perro, pero el de la clase padre (Animal) no se altera, esto se debe a que en Python solo se tiene en cuenta el nombre del método y no existe sobrecarga.