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).

Para modificar el mensaje de ruido sin alterar la clase Animal, se debe sobrescribe el método en la clase Pajaro.

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?

Si se define la clase Pez sin ningún constructor, método o atributo, heredará todo de su clase padre (Animal). Por lo tanto:

- Constructor: Usará el constructor de Animal, que requiere los argumentos nombre, edad, y raza.
- Métodos y atributos: Heredará todos los métodos y atributos definidos en Animal y su cadena de herencia (SerVivo).

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 el constructor de Animal, nombre y edad se asignan directamente, sin usar super(), lo que crea duplicación de atributos entre Animal y SerVivo. Para unificar los atributos, se debe modificar el constructor de Animal para usar super():

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()?

Cambio propuesto:

```
1 def setNombre(self, nombre):
2     super().setNombre(nombre)
3
4 def getNombre(self):
5     return super().getNombre()
6
```

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?

- **¿Es heredado?** Sí, las subclases heredan el atributo _totalCreados de la clase SerVivo.
- **¿Ocurre ocultación?** Sí, cuando se define _totalCreados en una subclase como Animal, se esta creando un nuevo atributo específico para esa subclase, que oculta el de SerVivo.

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

Sí, los métodos getTotalCreados definidos en las clases hijas (Animal, Gato, Pajaro, etc.) sobrescriben al método getTotalCreados de la clase padre (SerVivo). Esto sucede porque tienen el mismo nombre y están redefinidos en las subclases. En Python, la sobrescritura ocurre automáticamente cuando una subclase define un método con el mismo nombre que uno en la clase padre.

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

Una clase que herede de Persona obtendrá:

- Métodos y atributos públicos y protegidos de Persona.
- Atributos y métodos de SerVivo, ya que Persona hereda de esta clase

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

El método aduenarAnimal en Persona toma un argumento x, que se almacena en _animalAcargo y ejecuta el método ruido de x.

- **Requisitos del objeto:** x debe ser de una clase que implemente el método ruido (por ejemplo, Animal o una subclase como Gato o Perro).
- **¿Puede ser un objeto SerVivo?** No, porque ruido no está definido en SerVivo.

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

Se puede obtener la cantidad de seres vivos creados accediendo al atributo de clase _totalCreados o al método de clase:

```
1 print(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

El método definido en Perro sobrescribe al método getRaza en Animal, ya que tienen el mismo nombre. En Python, no existe sobrecarga como en Java; el último método definido con el mismo nombre prevalece. Si se necesita preservar el comportamiento del método en la clase padre, podemos invocarlo con super().