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).

-Agregaría el método ruido a la clase Pájaro y dentro de este haría las modificaciones deseadas.

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?

-Tendrá el constructor de su clase padre(Animal), los argumentos que recibe el constructor de su clase padre y tanto los atributos como otros métodos que posee su ´padre´.

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?

-Cuando la clase Animal hereda a la clase SerVivo, hereda los atributos nombre y edad definidos en SerVivo. Sin embargo, en el inicializador de Animal se están redefiniendo los atributos nombre y edad en lugar de reutilizar los que ya existen en SerVivo, al hacer eso, se provoca que solo uno de los atributos con el mismo nombre ocupe un espacio de memoria, pues son de instancia.

Para que los atributos sean el mismo en el inicializador, se debe hacer uso del inicializador de la clase SerVivo, llamándolo con el comando super().__init__(valores de los atributos que se deseen inicializar)

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()?

-Al definir los métodos, en lugar de generar retornos o establecimientos de valores propios de la clase Animal, hacer que retorne o establezca los valores del mismo método, pero de su clase padre.

Ejemplo:

```
 def setNombre(self, nombre):

super().setNombre(nombre)


def getNombre(self):

return super().getNombre()
```

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?

-El atributo _totalCreados definido en la clase SerVivo es heredado por todas sus clases hijas. Sin embargo, si en las clases hijas se redefine de, este atributo será ocultado en esas clases hijas, teniendo cada uno su atributo totalCreados propio de su clase.

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

-Si, ya que todos cuentan con la misma firma y tipo de retorno.

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

Hereda los métodos de la clase Persona y los que esta heredo de su clase padre(SerVivo).

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

Se le puede pasar un objeto de una clase que tenga un método ruido y si se le puede pasar un objeto SerVivo.

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

Retornando el valor del atributo de clase _totalCreados.

```
def __init__ (self, nombre, edad):
    self._nombre = nombre
    self._edad = edad
    SerVivo._totalCreados += 1
@classmethod
  def getTotalCreados(cls):
    return cls._totalCreados
```

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?

Cuando se cree un objeto de tipo Perro, hará uso del método getRaza de Perro. Este método se está sobrecargando, pues tiene una firma distinta a la de su clase padre.