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).

Sobrescribimos el método en la clase Pajaro, de la siguiente manera:

```
25
26      def ruido(self):
27          print("cantar y silbar")
28
```

En el archivo de 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?

Suponiendo que hereda de Animal:

En cuanto al constructor, no sobrescribe el método __init__ de Animal y por tanto recibe los argumentos nombre, edad y raza.

En cuanto a métodos y atributos:

- recibe todos los métodos y atributos de de Animal (_nombre,_edad, _raza y _totalCreados en atributos, setRaza, getRaza, setNombre, getNombre, caminar, correr, ruido y getTotalCreados como métodos) Así sean documentados como privados pues Python no tiene atributos privados.
- Además, tendría setEdad y getEdad de 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?

No es necesario cambiar el código del constructor para que esto pase, pues en el constructor de Animal, se hace referencia en _edad y _nombre al mismo espacio de memoria que referenciaría el constructor de serVivo. En cualquier caso, una modificación valida sería que, en el constructor de animal, pasáramos nombre y edad al constructor de serVivo en vez de que Animal se encargara de estos atributos, de la siguiente manera:

```
6       def __init__ (self, nombre, edad, raza):
7           super().__init__(nombre,edad)
8           self._raza = raza
9           Animal._totalCreados += 1
10
```

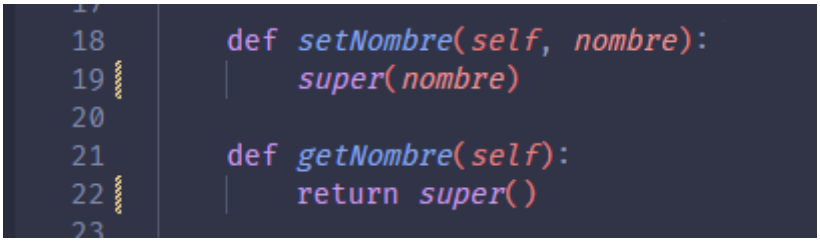
En Animal.

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()?

Podemos hacer que llamen a su contraparte de SerVivo de la siguiente manera:



en Animal.

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?

Debido a la falta de privacidad en Python, si se hereda, aunque en este código específicamente es siempre ocultado y debemos usar super o el nombre de la clase.

En este caso si vemos ocultación, pues si en Animal o cualquiera de sus subclases nos referimos _totalCreados, nos estaremos refiriendo al _totalCreados de la subclase en cuestión y para acceder a la de SerVivo debemos usar super, una cadena de supers o bien el nombre de la clase SerVivo.

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

Si, pues dada una instancia de Perro por ejemplo, podemos usar Perro.getTotalCreados y nos estaremos refiriendo a getTotalCreados de perro, no al definido en Animal.

7. ¿Qué métodos hereda una clase que hereda de la clase Persona?
    a. __init__ de Persona
    b. aduenarAnimal
    c. getTotalCreados
    d. setNmobre
    e. getNombre
    f. setEdad
    g. getEdad
8. ¿Qué tipo de objetos podrían pasársele al método aduenarAnimal de la clase Persona?¿Se le puede pasar un objeto serVivo?

No se le puede pasar un objeto meramente serVivo, pues un serVivo no necesariamente cuenta con el método ruido, y llamarlo produciría un error.

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

Usando 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 sobrescribe o se sobrecarga?

Es sobrescrito, y tratar de ejecutar getRaza en un perro sin argumentos pasaría a dar error.