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**).
   R/: En la clase Pajaro recrear el método ruido realizando un ocultamiento donde la herencia hará que tome este último método declarado
   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?
   R/: Esta clase tendrá por herencia el constructor de la clase Animal recibiendo parámetros nombre, edad, raza; como métodos tendrá setRaza, getRaza, setNombre, getNombre, caminar, correr, ruido, getTotalCreados y por cadena los que hereda Animal de la clase SerVivo setEdad, getEdad, y no hereda ningún atributo al estos ser “privados” esto indica que son propios de cada clase

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?
   R/: Para cada clase estos se asignan como nuevos atributos, ya que en la clase SerVivo son privados y por esto no se pueden heredar, existen únicamente para la misma clase; la forma para que fuesen el mismo seria quitarle la “privacidad” en ambas clases

4. En la clase Animal se sobrescribieron los métodos setNombre y getNombre, ¿Cómo modificaría estos métodos para que su funcionamiento no oculte algún valor de la clase padre? ¿Podría plantearse esta solución usando super()?
   R/: Para que estos atributos no se sobrescriban lo que deberíamos de hacer es en los métodos get y set de nombre en la clase animal agregarles el return super() para que estos realicen lo que este dentro del método en 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?
   R/: No es heredado por las demás clases por ser privado cada una tiene un espacio diferente de memoria reservado para este atributo y no ocurre la ocultación, pero si solo se toma que es de clase, si ocurre ocultación porque pertenece a cada clase

6. ¿Los métodos getTotalCreados sobrescriben al método de su padre?
   R/: Si sobrescribe, pero para la misma clase, haciendo que el método sea diferente para cada clase

7. ¿Qué métodos hereda una clase que hereda de la clase Persona?
   R/: Los métodos que heredaría serían los siguientes: aduenarAnimal, getTotalCreados, set y get de nombre y edad y el __init__ de la clase SerVivo

8. ¿Qué tipo de objetos podrían pasársele al método aduenarAnimal de la clase Persona? ¿Se le puede pasar un objeto serVivo?
   R/: Se le pueden pasar objetos de menor jerarquía en la cadena de herencia tal como: Animal o menor por ejemplo gato, perro etc. (se debe de tener en cuenta que el tipo del objeto tenga o herede el método ruido), y no se podría un objeto serVivo ya que es mayor en jerarquía y debe de ser menor o igual

9. ¿Cómo podría obtener la cantidad de seres vivos que se creen (**objetos creados**)?
   R/: Con la siguiente línea de código: **print(SerVivo.getTotalCreados), habiendo colocado un super().__init__() en la clase Animal**

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?

R/: Este método se estaría sobrescribiendo, volviendo este atributo como propio de su clase, el método funcionaria correctamente y retornaría la raza en este caso del perro