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).

- Simplemente habría que sobreescribir el método ruido para que este tenga el output esperado en la clase Pajaro. Ejemplo:

```
def ruido():
    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?

- Tomando en cuenta de que siguen el mismo comportamiento que el resto de las especies de animales, entonces al ser un hijo de Animal este heredaría absolutamente todos los métodos y atributos de está clase. Al ser python, el método init se hereda, entonces este sería el que tendría la clase.

```
Tabnine | Edit | Test | Explain | Document | Ask
def __init__ (self, nombre, edad, raza):
    self._nombre = nombre
    self._edad = edad
    self._raza = raza
    Animal._totalCreados += 1
```

Los **atributos** que se heredan son _nombre, _edad, _raza y _totalCreados.
Los **métodos** que se heredan serían los getters y setters de raza y nombre, caminar, correr y ruido.

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 la clase Animal, al definir de nuevo esos atributos que ya están en la clase SerVivo, digamos que la clase hija opaca a los de los de padre, haciendo que cada clase tenga su propia versión de los atributos. Para reutilizar los atributos de SerVivo, se debe usar super().__init__ en el constructor de la clase Animal.

```
Tabnine | Edit | Test | Explain | Document | Ask
def __init__ (self, nombre, edad, raza):
    super().__init__(nombre, edad);
    self._raza = raza
    Animal._totalCreados += 1
```

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()?

- Claramente se puede utilizar super() para poder acceder a los métodos de la clase padre. Así como en el código de abajo:

```
Tabnine | Edit | Test | Explain | Document | Ask
def setNombre(self, nombre):
    super().setNombre(nombre)

Tabnine | Edit | Test | Explain | Document | Ask
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?

- Sí se hereda totalCreados en las clases hijas, pero si se sobreescribe el atributo, cada clase en la que se sobreescriba se va a ocultar el atributo en la clase padre.

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

- Como había dicho anteriormente, cuando se sobrescribe algún método o atributo, esté oculta el de la clase padre, haciendo que cada hijo tenga su propia versión cuando se sobrescribe.

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

- Obtendrá el __init__ que es el método inicializador por defecto de la clase padre (Persona), el método aduenarAnimal() y el método de clase getTotalCreados().

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

- Se le podría pasar cualquier objeto siempre y cuando este contenga el método ruido, sino va a tirar un error al momento que quiera hacer el llamado a ruido. Entonces no se le podría pasar directamente el objeto SerVivo, a menos que se le añade el método ruido().

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

- Se puede usar el método getTotalCreados(). Eso devolverá el total de objetos creados en la clase SerVivo.

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?

- Teóricamente se sobrecarga, ya que hay un parámetro adicional que es tipo. (Hay que recordar que Python no permite sobrecargar entonces por eso digo teóricamente)