Stiven Santiago Rosero Quemag

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 del método ruido en la clase Pajaro sin alterar la clase Animal, debemos sobreescribir el método ruido en la clase Pajaro.

```
def ruido(self):
        return "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?

Si se crea una nueva clase Pez que hereda de la clase Animal sin definir nuevos métodos, constructor y atributos, la clase Pez heredará el constructor y los métodos de la clase Animal. El constructor de la clase Pez recibirá los mismos argumentos que el constructor de la clase Animal, que son nombre, edad y raza.

```
from Animal import Animal

class Pez(Animal):
   pass
```

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 se define la clase Animal que hereda de la clase SerVivo, los atributos nombre y edad de la clase SerVivo se heredan en la clase Animal. en el constructor de la clase Animal, se redefine el atributo nombre y edad se puede modificar el constructor de la clase Animal para que utilice los atributos heredados de la clase SerVivo. con super().__init_(nombre, edad)

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()?

Sí, se puede modificar los métodos setNombre y getNombre de la clase Animal para que utilizar la palabra clave super() para llamar a los métodos correspondientes de la clase padre SerVivo. De esta forma, se evita la ocultación de los valores de la clase padre.

```
   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?

Sí, el atributo totalCreados de la clase SerVivo es heredado por las demás clases que heredan de SerVivo. Sin embargo, al definir el atributo totalCreados de nuevo en las clases hijas, se produce una ocultación del atributo original de la clase SerVivo.

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

Sí, los métodos getTotalCreados de las clases hijas sobrescriben al método getTotalCreados de la clase padre SerVivo. Esto significa que cuando se llama al método getTotalCreados desde una clase hija, se ejecuta el método getTotalCreados de la clase hija y no del método getTotalCreados de la clase padre.

7. ¿Qué métodos hereda una clase que hereda de la clase Persona?
Una clase que hereda de la clase Persona hereda los siguientes métodos:
__init__
getTotalCreados
ruido

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

El método aduenarAnimal de la clase Persona, espera un objeto que tenga el método ruido(), por lo tanto se le puede pasar un objeto de una subclase de SerVivo que tenga un método llamado ruido().

9. ¿Cómo podría obtener la cantidad de seres vivos que se creen (objetos creados)?
Se puede obtener la cantidad de seres vivos que se creen utilizando el método getTotalCreados de la clase Persona. Este método devuelve el valor del atributo _totalCreados, que se incrementa cada vez que se crea un nuevo objeto de la clase Persona.

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 getRaza de la clase Animal se sobrecarga. La sobrecarga de métodos ocurre cuando se define un método con el mismo nombre que otro método, pero con una firma diferente (es decir, con un número o tipo de parámetros diferente).