Preguntas de analisis

Taller 5 python

Valentina sierra durango

1) 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// Para realizar esta modificacion lo que se debe hacer es sobrescribir el método ruido dentro de la clase Pajaro.

Como se muestra a continuacion:

```
class Pajaro(Animal):
  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?

tendrá el constructor de la clase Animal, recibe sus atributos y métodos de la 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?

Al definir nuevamente nombre y edad en Animal, se ocultan los atributos de la clase SerVivo.

Para evitar duplicación y mantener los mismos atributos, puedes usar el constructor de SerVivo con super() dentro de Animal:

```
class Animal(SerVivo):
  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, ¿Como modificaría estos métodos para que su funcionamiento no oculte algún valor de la clase padre? ¿Podría plantearse esta solución usando super()?

Para que los métodos no oculten el comportamiento de la clase padre, se puede usar super() para invocar los métodos correspondientes:

```
class Animal(SerVivo):
  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, _totalCreados es heredado por las clases hijas. Sin embargo, al redefinirlo en las clases hijas (Animal, Gato, etc.), se oculta el atributo de la clase padre. Cada clase tendrá su propia versión de _totalCreados, independiente de las demás.

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

Sí, los métodos getTotalCreados definidos en las clases hijas sobrescriben el método de la clase padre (SerVivo). Cada clase accede a su propia versión de _totalCreados.

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

Una clase que herede de Persona heredará todos los métodos y atributos de Persona, incluyendo:

aduenarAnimal

Métodos y atributos de SerVivo (como setNombre, getNombre, _totalCreados).

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

Se pueden pasar objetos de cualquier clase que implemente el método ruido, ya que aduenarAnimal llama a x.ruido().

No se puede pasar un objeto SerVivo directamente, a menos que SerVivo implemente el método ruido.

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

Llamando al método getTotalCreados de la clase SerVivo:

totalSeresVivos = 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 sobreescribe o se sobrecarga?

Este método sobreescribe al método getRaza de la clase Animal porque tiene el mismo nombre.

Sin embargo, cambia su firma al incluir un parámetro adicional (tipo), lo que hace que sea incompatible con llamadas esperadas por el método de la clase Animal.