Respuestas ejercicio 2 taller 5: Python

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/ Debería implementar un método `ruido` dentro de la clase `Pajaro`, el cual reemplace al de `Animal`. Esto sería algo como:
   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?
R/ Si la clase `Pez` se crea como una subclase de `Animal`, heredará todo lo de esta última, incluyendo el constructor, ya que este, al igual que otros métodos en Python, se hereda automáticamente. Aceptará los mismos parámetros, y los métodos y atributos serán los de la clase `Animal`.

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/ Los atributos `nombre` y `edad` están siendo redefinidos en la clase `Animal`. Para que estos atributos sean los mismos, habría que modificar el constructor de `Animal` para que pase estos parámetros al constructor de `SerVivo`. El cambio sería así:

```
class Animal(SerVivo):
    _totalCreados = 0
    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()?
R/ Sí, utilizando `super()` se puede acceder a los métodos de la clase base sin reemplazarlos. Para el caso de los métodos `getter` y `setter` de `nombre`, podría hacerse algo así:

```
def setNombre(self, nombre):
    super().setNombre(nombre)

def getNombre(self):
    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?
R/ Sí, este atributo es heredado por las subclases de `SerVivo` y `Animal`. Sin embargo, al redefinirlo en las clases hijas, se genera una ocultación del valor original del atributo.

6. ¿Los métodos getTotalCreados sobrescriben al metodo de su padre?
R/ Sí, esto ocurre porque en Python, los métodos de clase o métodos estáticos heredados se sustituyen al ser redefinidos, en lugar de ocultarse como sucede en Java.
7. ¿Qué métodos hereda una clase que hereda de la clase Persona?
R/ Heredará los métodos que `Persona` obtiene de `SerVivo`, como los `getters` y `setters` de `nombre` y `edad`, y el método `getTotalCreados`. También hereda los métodos de la clase `Object` como `__str__`, además de los métodos propios de `Persona`, como `aduenarAnimal`.
8. ¿Qué tipo de objetos podrían pasársele al método aduenarAnimal de la clase Persona?¿Se le puede pasar un objeto serVivo?
R/ Solo se pueden pasar objetos de tipo `Animal`, es decir, instancias de la clase `Animal` o de sus subclases, ya que estas tienen el método `ruido()` definido. Si se pasa un objeto de tipo `SerVivo`, al no contar con el método `ruido`, se generará un error: `AttributeError: 'SerVivo' object has no attribute 'ruido'`
9. ¿Cómo podría obtener la cantidad de seres vivos que se creen (**objetos creados**)?
Si las subclases de `SerVivo` y `Animal` llaman al constructor de su clase base mediante `super()`, todos los objetos creados terminarán invocando el constructor de `SerVivo`, donde se incrementa un contador. Por lo tanto, para obtener el número total de seres vivos creados, basta con usar el método `getTotalCreados()` de 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 sobreescribe o se sobrecarga?
R/ El método se redefine. En Python no existe la sobrecarga.