Respuestas taller 5 Python

1. Basta con sobrescribir el método ruido dentro de la clase Pajaro para así no tener que alterar la clase padre Animal. Algo más o menos de la siguiente manera:

```
    def ruido(self):
        print("Cantar y silbar")
```

2. Suponiendo que la nueva clase Pez hereda de Animal y no le definimos nada, en teoría debería de heredar absolutamente todos los atributos y métodos de la clase Animal. Incluso se hereda el inicializador, que es una característica bastante peculiar de Python.
3. Para que los atributos sean los mismos en la clase SerVivo y la clase Animal, basta con añadirle el super al inicializador ya que este se encarga de inicializar los atributos de la clase padre con base en estos parámetros que le pasemos. Sería algo más o menos así:

```
    def __init__ (self, nombre, edad, raza):
        super().__init__(nombre, edad)
        self._raza = raza
        Animal._totalCreados += 1
```

4. Basta con añadir un super dentro de los métodos setNombre y getNombre para así llamar a los métodos de la clase padre: por ejemplo:

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

    def getNombre(self):
        return super().getNombre()
```

5. En teoría los atributos de clase si pueden ser heredados. Sin embargo, en este caso como estas definiendo de nuevo el atributo dentro de las clases hijas, entonces ocurre una ocultación en donde se crea un nuevo atributo para cada subclase que no será compartido por ellas y su clase padre.
6. En Python como tal si se sobrescriben los métodos estáticos a diferencia de Java que ahí se ocultan. Por lo tanto, los métodos getTotalCreados si sobrescriben al método de la clase padre en Python.
7. Si creamos una clase que hereda de Persona, esta clase hereda todos los métodos relacionados con Persona (incluyendo el inicializador) y a su vez hereda los métodos que haya en SerVivo ya que Persona hereda de SerVivo.
8. En teoría se puede pasar un objeto de SerVivo al método aduenarAnimal de la clase Persona debido al Polimorfismo de Python, sin embargo, en este caso ocurre un error ya que la clase SerVivo no tiene definido un método ruido() por lo cual si le pasas un objeto de tipo SerVivo (que es el único que no tiene este método junto a Persona) saldrá un error.

9. Para obtener la cantidad de seres vivos o la cantidad de objetos creados basta con llamar al método getTotalCreados() de la clase SerVivo, este método accede y devuelve el valor que hay asignado en el atributo _totalCreados. Este atributo se aumenta en el inicializador de la clase SerVivo, esto asegura que cada vez que se crea una nueva instancia de SerVivo, entonces aumenta el valor de _totalCreados.
10. Técnicamente si defines este nuevo método en la clase Perro:

```
    def getRaza(self, tipo):
        return self._raza + " " + tipo
```

    Se esta sobrecargando el método getRaza de la clase Animal porque la firma del nuevo método es completamente distinta a la del método original, por lo cual no puede haber una sobrescritura.