Ejercicio 2 Taller 5 Python

Tomas Aristizabal Gomez - 1033181207

1. Para ello se debe sobreescribir el método **ruido()** de la clase **Animal** en la clase **Pajaro** asi:

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

2. Si se crea una clase **Pez** que hereda de **Animal,** esta tendrá el constructor de su clase padre, que recibe un nombre, una edad y una raza. Además tendrá los respectivos métodos definidos en la clase **Animal** y **serVivo** (padres de esta).
3. Los atributos **nombre** y **edad** están siendo duplicados, debido a que no se está haciendo un llamado al constructor de la clase padre (**serVivo**). La solucion seria llamar a ese constructor usando **super()** asi:

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

4. Para evitar ocultar algun valor de la clase padre, se podría hacer lo siguiente:

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

Asegurando que se está llamando al método desde la clase padre y sin ocultar algún valor

5. Si, el atributo **totalCreados** es heredado por las clases hijas y todas comparten el mismo valor. En caso de de que se defina ese atributo en una clase hija, se creará una nueva variable de clase para esa clase hija específicamente, no es precisamente una ocultación sino que esa clase usará su propia versión del atributo pero podrá seguir accediendo a él de su clase padre
6. Si, los métodos sobreescriben a el creado en su clase padre, ya que al llamarlos usan su propia versión del método, en este caso retornar **totalCreados** bajo su contexto
7. Una clase que hereda de la clase persona, heredará los métodos definidos en la clase **Persona** y ademas los definidos en la clase **serVivo,** en caso de que algun metodo este sobrescrito, se tomará el de el pariente más cercano, en este caso el de la clase **Persona**
8. Se le podría enviar objetos de tipo **Animal** y todas sus clases heredadas, con la condición de que ese objeto tenga el método **ruido().** Implícitamente al pasarle objetos de tipo **Animal,** es también de tipo **serVivo,** por ende si es posible, pero con la condición dada.
9. mediante la siguiente línea: `SerVivo.getTotalCreados()` para acceder a la cantidad “general” de seres vivos creados

10. Se está **sobreescribiendo** a el método **getRaza** de la clase padre, osea que se reemplaza cuando es llamado desde una instancia de la clase hija. Recordar que la sobrecarga como tal no existe en python.