Respuestas

1) Puedes sobrescribir el método ruido en la clase Pajaro:

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

2) La clase Pez heredará todo de Animal. Tendrá el constructor de Animal:

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

También heredará los métodos de Animal como:

setNombre, getNombre
setRaza, getRaza
caminar, correr, ruido
El método de clase getTotalCreados

3) Los atributos _nombre y _edad son redefinidos en Animal pero no son compartidos con los de SerVivo. Para que ambos sean los mismos, deberías llamar al constructor de la clase padre usando super():

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

4) Puedes utilizar super() dentro de los métodos sobrescritos para invocar los métodos de la clase padre:

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

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

5) Sí, totalCreados es heredado, pero si se redefine en las subclases (Animal, Pajaro, etc.), se oculta. Cada subclase tendrá su propia versión de _totalCreados.

6) Sí, los métodos getTotalCreados en las subclases sobrescriben al de SerVivo.

7) Cualquier clase que herede de Persona tendrá los métodos de Persona (aduenarAnimal, getTotalCreados) y los de SerVivo (setNombre, getNombre, etc.).

8) aduenarAnimal puede recibir cualquier objeto que tenga el método ruido, como un Animal, Pajaro, Perro, etc. No puede recibir directamente un SerVivo, ya que SerVivo no tiene el método ruido.

9) Llamando al método getTotalCreados de la clase SerVivo:

   print(SerVivo.getTotalCreados())

10) El método getRaza en Perro sobrescribe al de Animal. No se sobrecarga porque Python no soporta sobrecarga directa de métodos.