

1) Debes sobrescribir el método ruido dentro de la clase Pájaro. Por ejemplo:

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

2) Tendrá el constructor de la clase Animal, recibe sus atributos y métodos de la clase

2) 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) 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) 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) 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) 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) 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) Llamando al método `getTotalCreados` de la clase `SerVivo`:

`totalSeresVivos = SerVivo.getTotalCreados()`

10) Este método sobrescribe 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`.