Respuestas Taller 5 - Python.

1. Se puede definir un método en la clase Pajaro el cual tenga la siguiente estructura

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

Esto lo que hace es un polimorfismo del método ruido que estaba en Animal. Si se declara un objeto Pajaro y se pide un print de este método, el devolvera “cantar y silbar”

2. Asumiendo que la clase Pez es una clase hija de la clase Animal, está heredará el constructor que está en Animal, que recibe los argumentos nombre, edad y raza, mientras que su atributo será _totalCreados. La clase Pez también heredará los métodos definidos en la clase Animal, incluido el método de clase. Los métodos podrán acceder a los atributos nombre, edad y raza si usan un self, asimismo, podrán usar a _totalCreados si usan cls.

3. Los atributos nombre y clase son ahora inicializados por Animal, en vez de usar los que estaban definidos desde un comienzo en la clase SerVivo, para que estos compartan el mismo inicializador, podemos hacer que Animal herede el inicializador de SerVivo con las siguientes líneas

```
 def __init__(self, nombre, edad, raza):
      super().__init__(nombre, edad)  # Call SerVivo's __init__
      self._raza = raza
```

super(). Hará un llamado a la superclase SerVivo donde usará el inicializador que se había definido allí. Es necesario definir el atributo raza puesto que este no está en el inicializador de SerVivo, pero ahora la clase Animal usará el inicializador de SerVivo para definir parte de sus atributos.

4. Para que no se oculte el valor de la clase padre, basta con modificar el codigo en la clase Animal de la siguiente forma:

```
def setNombre(self, nombre):
      #self._nombre = nombre
      super().setNombre(nombre)

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

Esto asegura que se están llamando los valores de la clase padre. Independientemente de que un constructor vaya a sobreescribirlos o usarlos, esto asegura que el “nombre” en la clase padre no esté siendo ocultado.

5. Si, este atributo es heredado por las demás clases hijas, pero es reescrito en cada una de ellas, lo que hace que este valor se oculte y dependerá de la referencia para obtener dicho valor, dependiendo si dicha referencia apunta hacia el totalCreados de la clase padre o hija.

6. Si, el método getTotalCreados es sobreescrito en todas las clases hijas que tengan definido el método ya que Python sobreescribe toda clase de métodos cuando hablamos de herencia. Por ejemplo, si creamos un objeto en la clase SerVivo, el contado subirá a 1, pero al crear dos objetos de la clase Animal, esté solo contara los objetos creados en la clase Animal, es decir, el contador solo mostrará un 2, pero la clase SerVivo tendrá como resultado 3.

7. Supongamos que esta clase se llama “Adulto”, que es una clase hija de la clase Persona. La clase Adulto heredará todos los métodos que están ubicados en la clase Persona, pero esta misma está heredando los métodos de la clase SerVivo, por lo tanto, la clase Adulto hereda los métodos definidos tanto en la clase Persona como en la clase Animal y la clase SerVivo, teniendo en cuenta que algunos de los métodos de la clase SerVivo podrían haber sido reescritos en la clase Persona.

8. No, no se le puede pasar un tipo SerVivo. Para que este método funcione, el objeto debe pertenecer a una clase que tenga el método “ruido” definido, de lo contrario va a arrojar un error al correr el código, por lo tanto, el tipo de objeto que puede pasar a traves del metodo aduenaAnimal son Gato, Pajaro, Perro. En el caso de Animal, esta tiene definida un método ruido, pero no tiene retorno sino un pass.

9. Como se mencionó con antelación, el contador _totalCreados en SerVivo toma en cuenta todos objetos creados en su propia clase y en sus clases hijas, por lo tanto, el mejor modo de obtener cuantos animales han sido creados en el programa es ejecutar el siguiente método

```
SerVivo.getTotalCreados()
```

10. Este se sobreescribe. A diferencia de Java, aquí no hay sobrecarga, por lo tanto, el método getRaza en Perro sobreescribe el metodo getRaza en Animal gracias a el modo en el cual Python ejecuta su código.