1. Se debe sobrescribir el método ruido() en la clase Pajaro, así:

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

2. Esta nueva clase Pez, al ser un hijo de Animal, heredaría todos sus métodos y atributos, incluyendo el __init__.
3. Los atributos nombre y edad de la clase serVivo son redefinidos en la clase Animal, cuando se crea un objeto de tipo Animal. Para cambiar el código del inicializador de manera que estos atributos sean el mismo, debemos redefinir el constructor de la clase Animal de la siguiente forma:

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

   Con esto, se le envían los atributos nombre y edad a los atributos definidos en su clase padre (clase serVivo), lo que hace que estas dos clases tengan los mismos valores en sus atributos nombre y edad.
4. Sí, de la siguiente manera:

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

5. totalCreado si se hereda, pero este se va a ver ocultado en cada clase en la que se redefine
6. Sí, ya que el método getTotalCreados() está definido con el mismo nombre en cada clase hija de la clase serVivo (Persona, Animal, Gato, Pajaro, Perro), sobrescribiendo el método de la clase padre. Esto hace que cada subclase retorne solo su propio contador.
7. De la clase Persona se heredan todos los métodos.
8. Al método aduenarAnimal se le debería poder pasar todo objeto que posea el método ruido(), en este caso sería Animal y sus hijos, dejando por fuera a SerVivo.

9. Para obtener la cantidad total de objetos, se debe llamar al metodo getTotalCreados() desde SerVivo.
10. En este caso el getRaza de Perro sobreescribirá al getRaza de Animal.