Test

Ejercicio 1

En Ruby, se pueden utilizar métodos consultores dentro de clases dependiendo del orden en que se inicialicen los atributos.

```
class Persona {
    String nombre;
    String apellidos;
    String nombre_completo;
}
```

Solución 1: no está mal pero podría ser mejor

```
Persona (String n, String a){
   nombre = n;
   apellidos = a;
   nombre_completo = a + n;
}
```

Solución 2: no es una solución muy acertada porque n y a pueden cambiar

```
Persona (String n, String a){
   setNombre(n);
   setApellidos(a);
   nombre_completo = n + " " + a;
}
```

Solución 3: esta solución es muy válida

```
Persona (String n, String a){
   setNombre(n);
   setApellidos(a);
   nombre_completo = getNombre() + " " + getApellidos();
}
```

Solución 4:

```
Persona (String n, String a){
   setNombre(n);
   setApellidos(a);
   nombre_completo = nombre + " " + apellidos;
}
```

Ejercicio 2

Se puede hacer this.setNombre(n) dentro de un constructor.

Ejercicio 3

No hace falta crear un método de modificador exclusivamente para realizar una comprobación, por lo que se utiliza attr_accesor + comprobación.

Ejercicio 4

En el código:

```
p1 = new Persona("Luis");
p2 = p1;
p3 = p2;
```

Como tal hay un objeto porque solo hay un new, p2 y p3 son variables apuntando al mismo objeto. Por lo que si en p2 cambio algo del objeto, afecta a p1 y p3 porque es el mismo objeto.

EJERCICIO CLASE

Crear en Ruby una clase "Perro" que reciba por parámetro en el constructor el nombre y el ID sea automático.

```
module Basico
  class Perro
    @@contador = 0
    attr_reader :nombre

    def initialize(nombre)
      @@contador += 1
      #inicializo las variables
      @nombre = nombre
      @id = @@contador
    end

    def getId()
      puts @id
    end

  end

  p1 = Perro.new("Perro 1")
  p2 = Perro.new("Perro 2")

  p1.getId()
  p2.getId()
end
```

En Ruby los atributos siempre son privados para acceder a ellos y entonces lo que se hizo fue implementar consultores y modificadores.

- Ahora modificamos el código para que nadie llame "Perro" a su perro.

Solución 1

```
module Basico
  class Perro
    @@contador = 0
    attr_reader :nombre

    def initialize(nombre)
      @@contador += 1
      #inicializo las variables
      if(nombre != "Perro")
        @nombre = nombre
      end
      @id = @@contador
    end

    def getId()
      puts @id
    end

  end

  p1 = Perro.new("Perro 1")
  p2 = Perro.new("Perro 2")

  p1.getId()
  p2.getId()
end
```

Solución 2:

```
module Basico
  class Perro
    @@contador = 0
    attr_accessor :nombre

    def initialize(nombre)
      @@contador += 1
      #inicializo las variable
      self.nombre = nombre
      @id = @@contador
    end

    def getId()
      puts @id
    end

    def nombre=(n)
      if(n != "Perro")
        @nombre = n
      end
    end
```

```
  end

  p1 = Perro.new("Perro 1")
  p2 = Perro.new("Perro 2")

  p1.getId()
  p2.getId()
end
```