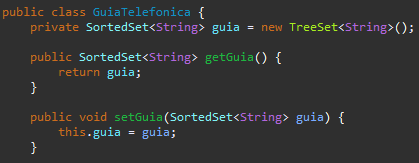
**Refactor 1**

En GuiaTelefonica

Mal olor: Declaración de atributo público.

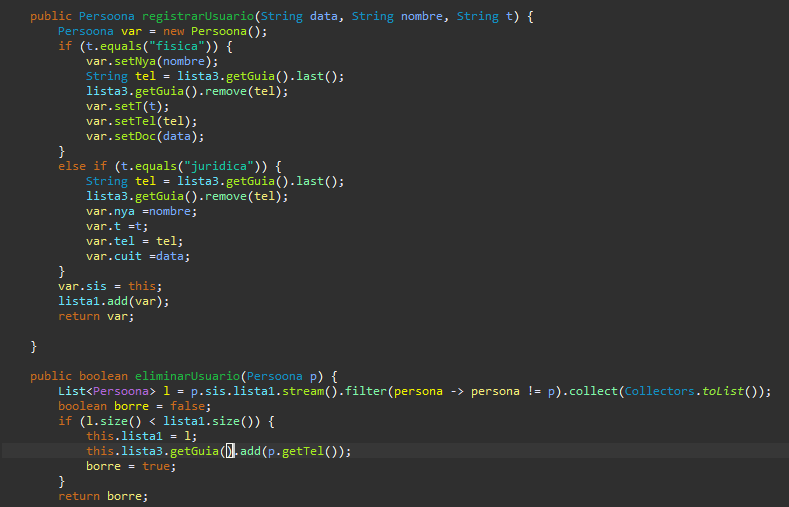


Refactoring a aplicar: Encapsulate Field



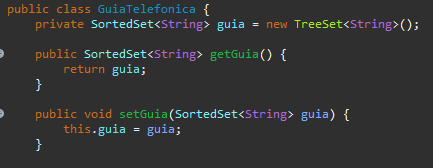
En class Persoonal:



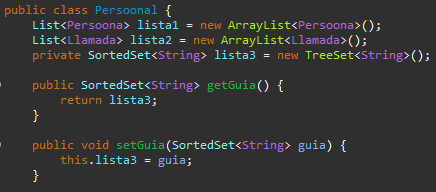


**Refactor 2**

Mal olor: Data Class



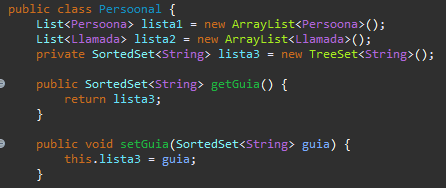
Refactoring a aplicar: Move Method

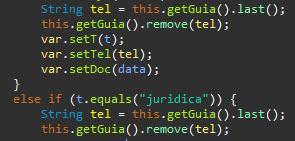


Código con el refactoring aplicado:

Se eliminó clase GuiaTelefonica (quedaba vacía)(Lazy Class???)

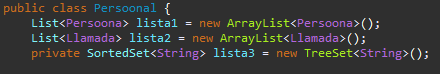
Clase Persoonal:





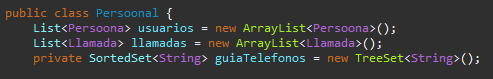
a) Mal olor detectado en el código: los nombres de atributos no son claros (Clase Persoonal)

b) Extracto del código que presenta el mal olor:



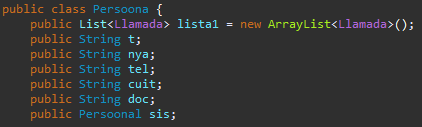
c) Refactoring a aplicar que resuelve el mal olor: cambio de nombres de atributos en Clase Persoonal

d) Código con el refactoring aplicado



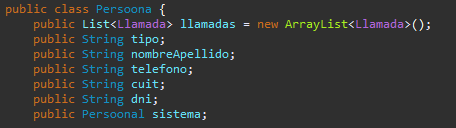
a) Mal olor detectado en el código: los nombres de atributos no son claros (Clase Persoona)

b) Extracto del código que presenta el mal olor:



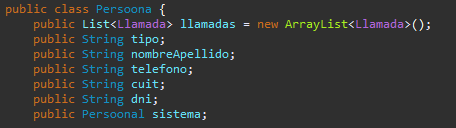
c) Refactoring a aplicar que resuelve el mal olor: renombrar atributos

d) Código con el refactoring aplicado



a) Mal olor detectado en el código: Declaración de atributo público (en Persoona)

b) Extracto del código que presenta el mal olor:

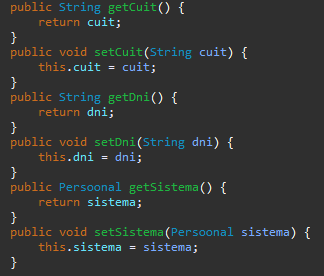


c) Refactoring a aplicar que resuelve el mal olor: Encapsulate Field

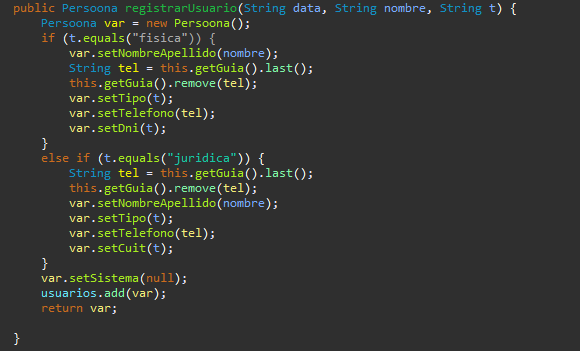
Ademas renombro los getters y setters correspondientes

d) Código con el refactoring aplicado





Personaal:



a) Mal olor detectado en el código: Dead Code

b) Extracto del código que presenta el mal olor:



(es privado y ni siquiera tiene getter)

c) Refactoring a aplicar que resuelve el mal olor

(borrarlo? )

d) Código con el refactoring aplicado ¿lo borro?

a) Mal olor detectado en el código: el nombre del atributo no es claro (Clase Llamada)

b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor: renombrar atributo

d) Código con el refactoring aplicado



a) Mal olor detectado en el código: Declaración de atributo público (Clase Llamada)

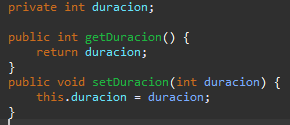
b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor: Encapsulate Field

d) Código con el refactoring aplicado

Llamada:

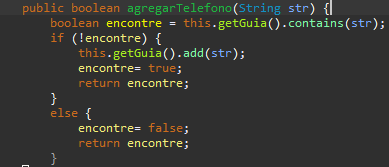


Persoonal:



a) Mal olor detectado en el código: speculative generality (Clase Persoonal) (o Dead Code?? )

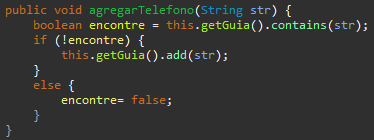
b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor

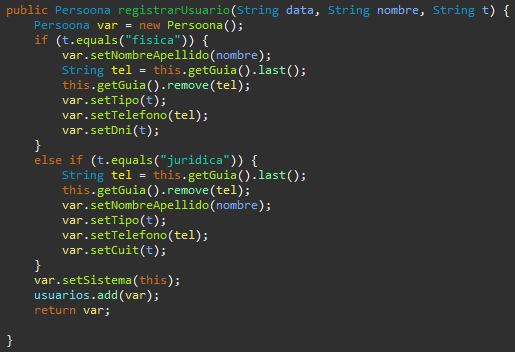
Remover valor de retorno del método sin uso

d) Código con el refactoring aplicado



a) Mal olor detectado en el código: nombres de parámetros y variables de método registrarUsuario no descriptivos. (Clase Persoonal)

b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor: renombrar parámetros y variables del método registrarUsuario

d) Código con el refactoring aplicado

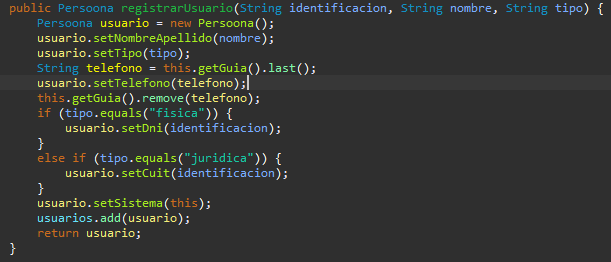
a) Mal olor detectado en el código: Long Method

b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor: Consolidate Duplicate Conditional Fragments

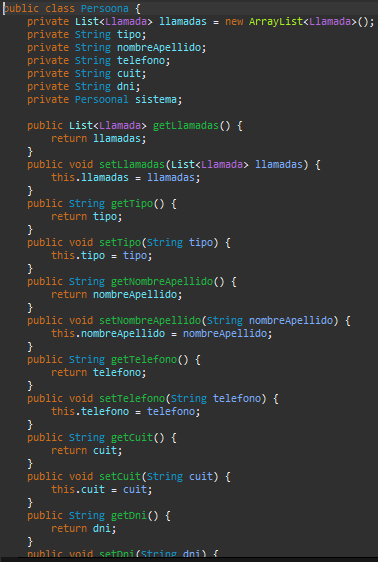
d) Código con el refactoring aplicado

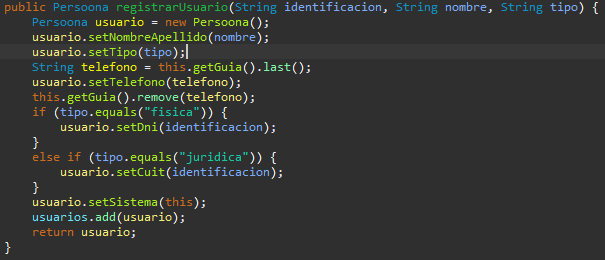


a) Mal olor detectado en el código: Large Class (Persoona),Long Method (Persoonal)

La clase persona tiene campos que son para un tipo específico de persona.

Además se debería usar un constructor para inicializar sus atributos.

b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor:

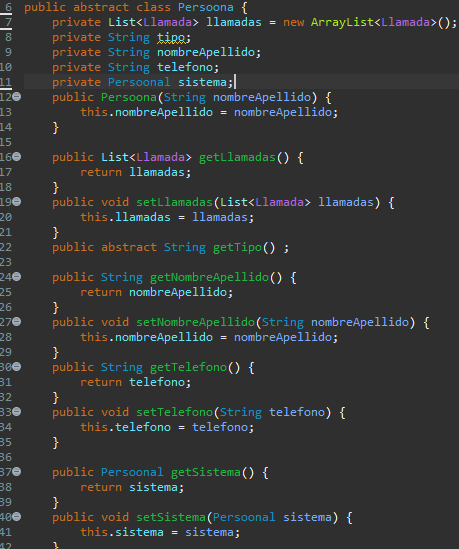
Extract Subclass

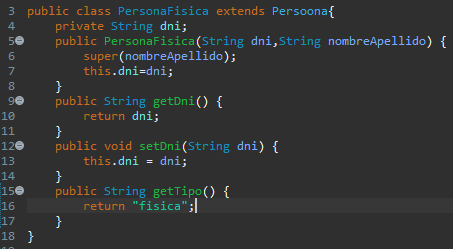
Push Down Field

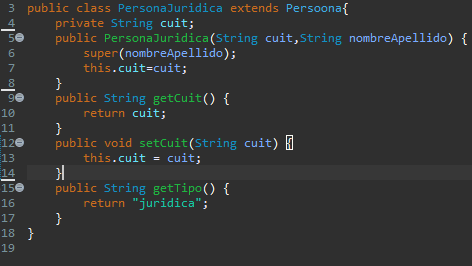
Push Down Method (getters y setters)

Hice abstracta a la clase persona ya que no tiene necesidad de ser instanciada.

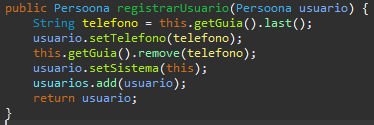
d) Código con el refactoring aplicado



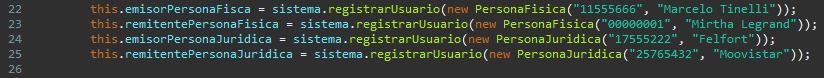




En Persoonal



En setup de los tests:



a) Mal olor detectado en el código:

b) Extracto del código que presenta el mal olor:

c) Refactoring a aplicar que resuelve el mal olor

d) Código con el refactoring aplicado

a) Mal olor detectado en el código:

b) Extracto del código que presenta el mal olor:

c) Refactoring a aplicar que resuelve el mal olor

d) Código con el refactoring aplicado

a) Mal olor detectado en el código:

b) Extracto del código que presenta el mal olor:

c) Refactoring a aplicar que resuelve el mal olor

d) Código con el refactoring aplicado

a) Mal olor detectado en el código:

b) Extracto del código que presenta el mal olor:

c) Refactoring a aplicar que resuelve el mal olor

d) Código con el refactoring aplicado

a) Mal olor detectado en el código:

b) Extracto del código que presenta el mal olor:

c) Refactoring a aplicar que resuelve el mal olor

d) Código con el refactoring aplicado