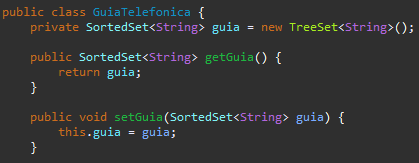
**Refactor 1**

En GuiaTelefonica

Mal olor: Declaración de atributo público.

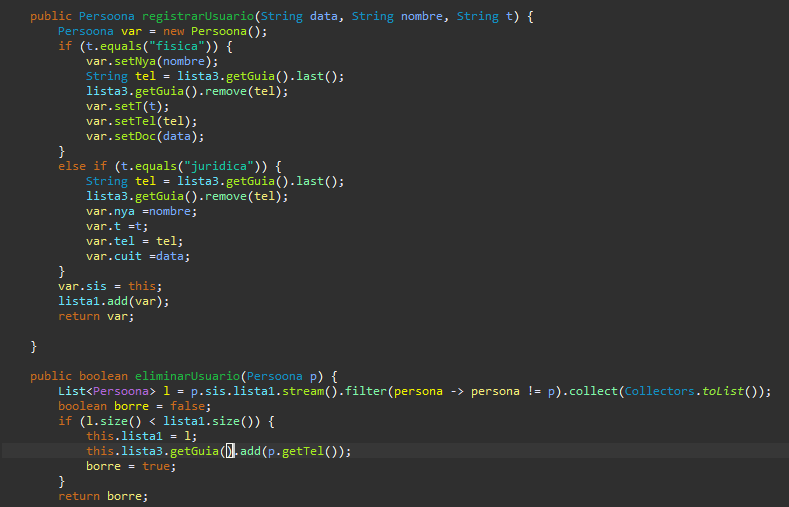


Refactoring a aplicar: Encapsulate Field



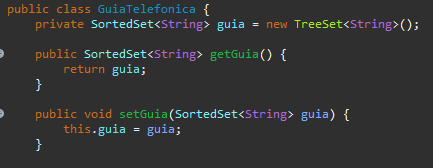
En class Persoonal:



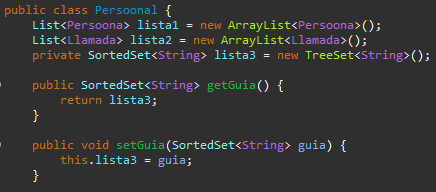


**Refactor 2**

Mal olor: Data Class



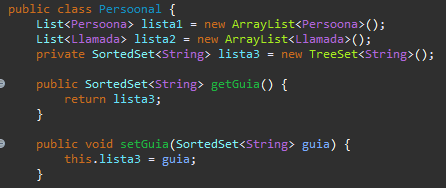
Refactoring a aplicar: Move Method

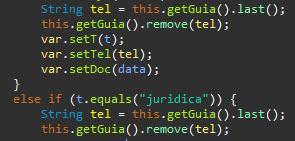


Código con el refactoring aplicado:

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

Clase Persoonal:

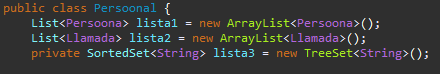




**Refactor 3**

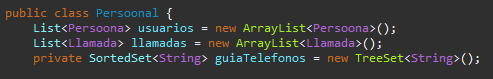
a) Mal olor detectado en el código: los nombres de atributos no son claros (Clase Persoonal)

b) Extracto del código que presenta el mal olor:



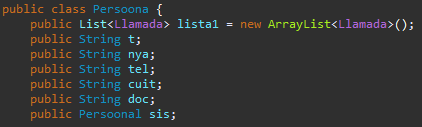
c) Refactoring a aplicar que resuelve el mal olor: cambio de nombres de atributos en Clase Persoonal

d) Código con el refactoring aplicado



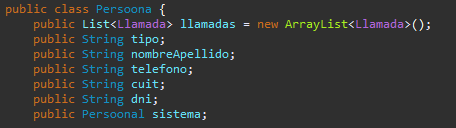
a) Mal olor detectado en el código: los nombres de atributos no son claros (Clase Persoona)

b) Extracto del código que presenta el mal olor:



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

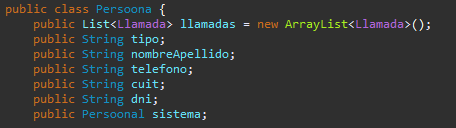
d) Código con el refactoring aplicado



**Refactor 4**

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

b) Extracto del código que presenta el mal olor:

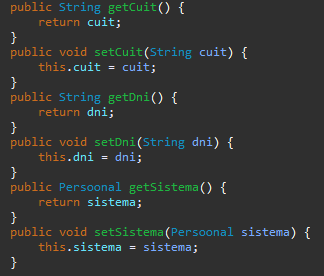


c) Refactoring a aplicar que resuelve el mal olor: Encapsulate Field

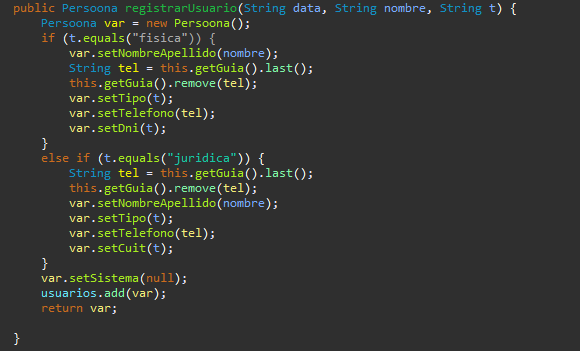
Ademas renombro los getters y setters correspondientes

d) Código con el refactoring aplicado





Personaal:



**Refactor 5**

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?

**Refactor 6**

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



**Refactor 7**

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

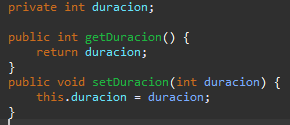
b) Extracto del código que presenta el mal olor:



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

d) Código con el refactoring aplicado

Llamada:



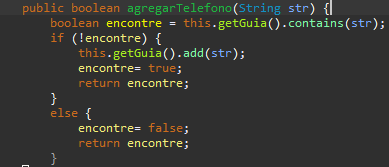
Persoonal:



**Refactor 8**

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

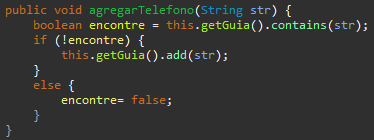
b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor

Remover valor de retorno del método sin uso

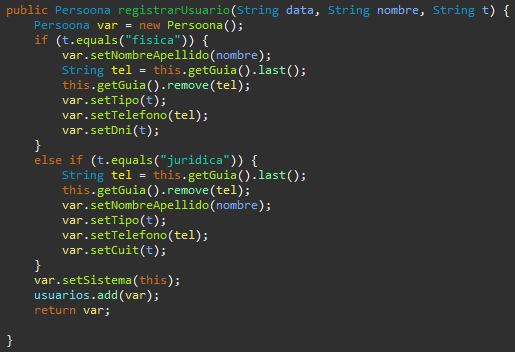
d) Código con el refactoring aplicado



**Refactor 9**

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

**Refactor 10**

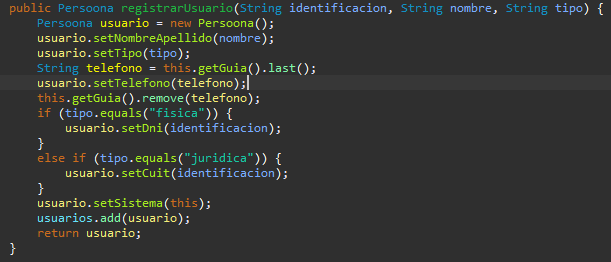
a) Mal olor detectado en el código: Long Method

b) Extracto del código que presenta el mal olor:



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

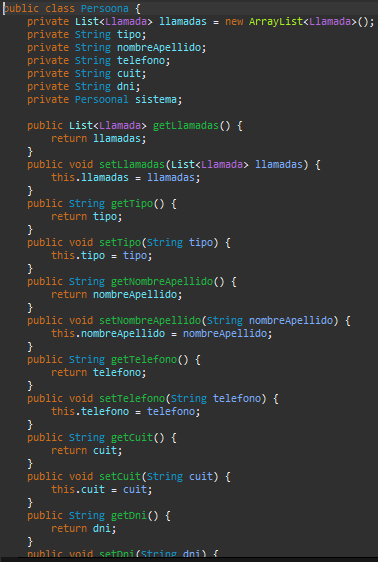
d) Código con el refactoring aplicado

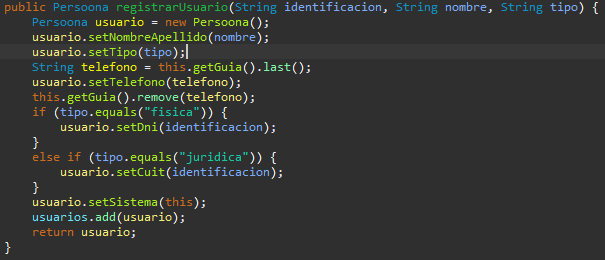


**Refactor 11**

a) Mal olor detectado en el código: Long Method (RegistrarUsuario en Persoonal),Switch Statements.

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

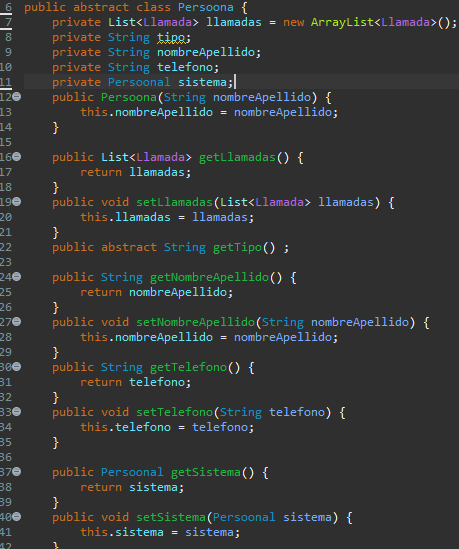
Push Down Field

Push Down Method (getters y setters)

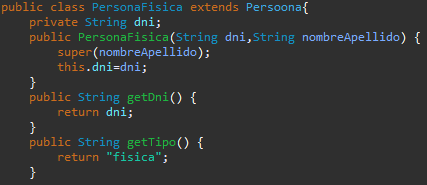
Hice abstracta a la clase persona ya que no tiene necesidad de ser instanciada.

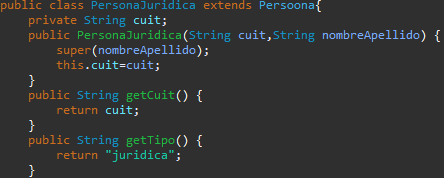
Eliminé setters que quedaron innecesarios por el uso de un constructor.

d) Código con el refactoring aplicado

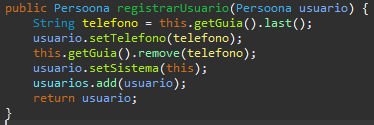




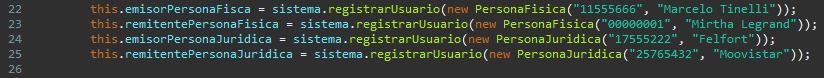




En Persoonal



En setup de los tests:



**Refactor 12**

a) Mal olor detectado en el código: reinventar la rueda al buscar si existe el usuario, Long Method (CalcularMontoTotalLlamadas en Persoonal),

b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor

Utilizar las herramientas que me brinda el lenguaje, como los streams.

Asignarle a la clase Persoona la responsabilidad de ver si una persona posee el mismo teléfono que la recibida por parámetros mediante el método “tieneTelefono”.

d) Código con el refactoring aplicado



**Refactor 13**

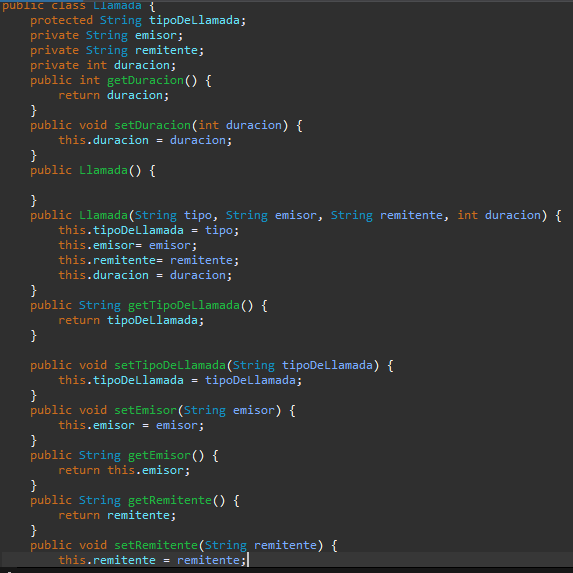
a) Mal olor detectado en el código: Nombres de atributos y parámetros no descriptivos dentro de la clase Llamada. Falta getter de emisor.

b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor: cambio de nombres de atributos y parámetros para hacerilos mas descriptivos

d) Código con el refactoring aplicado



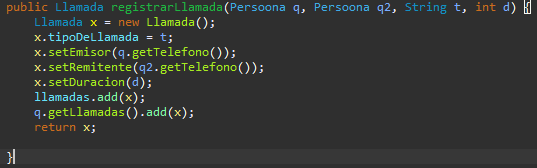
**Refactor 14**

a) Mal olor detectado en el código: Switch Statements (pregunta por tipos de llamadas), Long Method (CalcularMontoTotalLlamadas en Persoonal). La clase Llamada almacena un tipo.

En RegistrarLlamada no se el constructor de la clase Llamada para crear una instancia de llamada.

b) Extracto del código que presenta el mal olor:





c) Refactoring a aplicar que resuelve el mal olor: Replace Conditional with Polymorphism

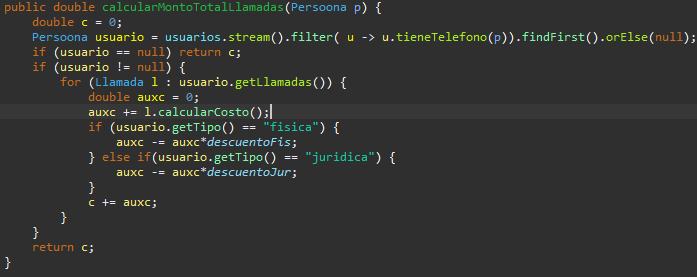
Modificar método “RegistrarLlamada” para que reciba directamente la instancia de la llamada.

Crear jerarquía de Llamadas y hacer método abstacto “calcularCosto()”.

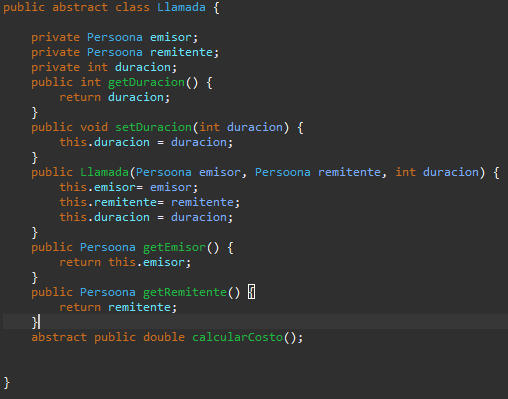
-------Completar ¿?

d) Código con el refactoring aplicado

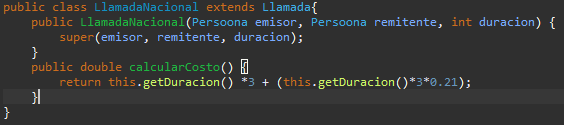
Clase Persoonal:



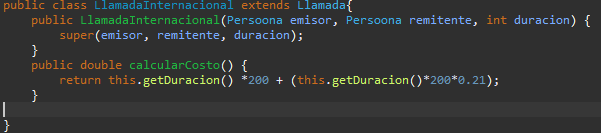
Clase llamada:

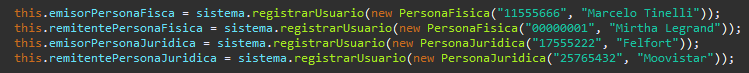


Nueva clase LlamadaNacional:



Nueva clase LlamadaInternacional:

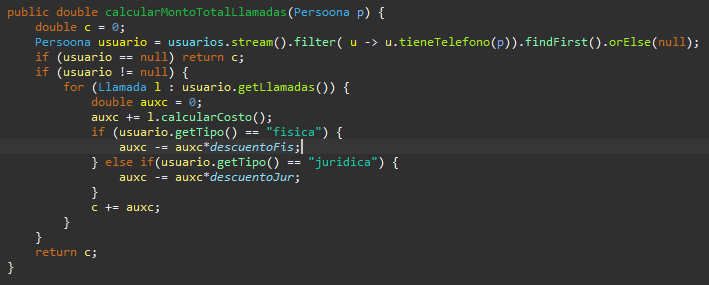


Setup de Tests: 

**Refactor 15**

a) Mal olor detectado en el código: Switch Statements en Método “calcularMontoTotalLlamadas”. (pregunta por tipo de persona), nombres de variables locales poco descriptivos (c, auxc,l)

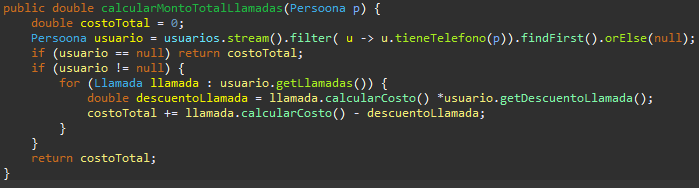
b) Extracto del código que presenta el mal olor:



c) Refactoring a aplicar que resuelve el mal olor: Replace Conditional with Polymorphism

d) Código con el refactoring aplicado

En clase Persoonal:



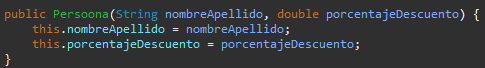
Eliminé en Persoonal:



En Persoona (clase abstracta) eliminé el atributo “tipo” y agregué el atributo porcentajeDescuento junto con su getter.:







En PersonaJurídica y PersonaFisica eliminé el getter de “tipo” y mediante el constructor de su superclase le asigno el valor al atributo “porcentajeDescuento”.





a) Mal olor detectado en el código:

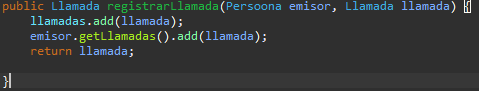
Responsabilidades mal asignadas: la persona debería tener registro de sus llamadas, no el sistema. Además la persona debería encargarse de realizar el calculo de calcular el monto de sus llamadas, no el sistema.

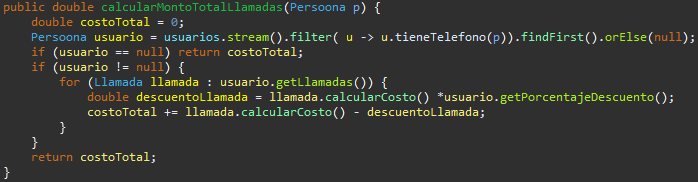
b) Extracto del código que presenta el mal olor:

En clase Persoonal:



(en la siguiente captura hay otro mal olor del cual me encargaré en otro refactor)





c) Refactoring a aplicar que resuelve el mal olor

Move Method.

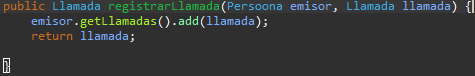
Eliminé la lista de llamadas de la clase de sistemas porque no era su responsabilidad tenerla.

d) Código con el refactoring aplicado

Clase Persoonal:

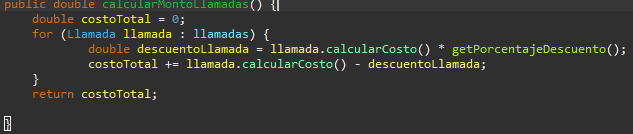


(en registrarLlamada eliminé la línea que registraba una llamada en la lista de llamadas de la clase Persoonal, se que sigue habiendo otro mal olor)





En Persoona:



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