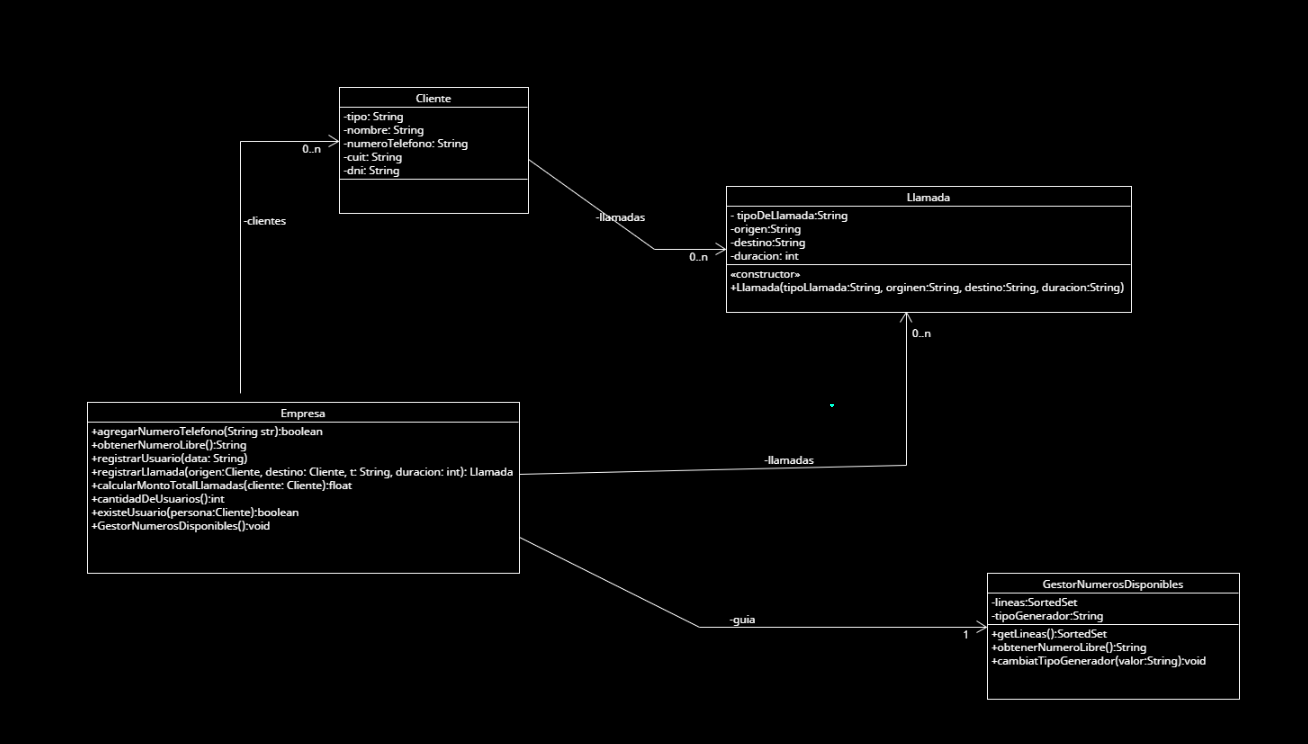
Ejercicio 3 - Facturación de llamadas

## Integrantes del grupo:

### -Francisco Manuel Jorge

### -Lautaro Gutierrez

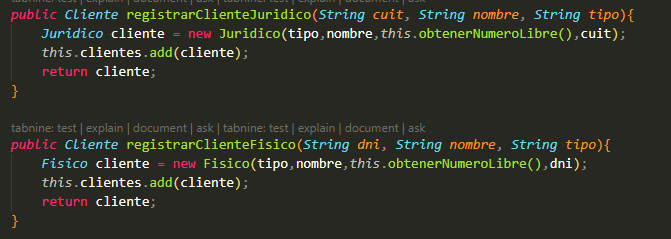
**-Diagrama UML con la solución provista**



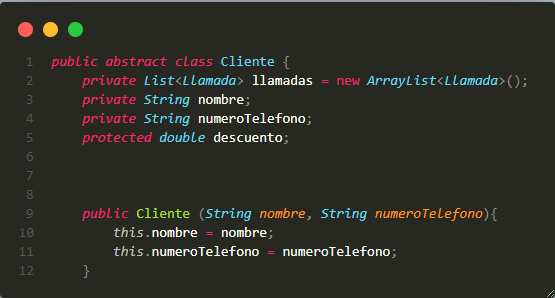
# 1- Mal olor detectado: “Switch statements”

Al tener ifs dentro del método para instanciar objetos según el tipo en registrar usuario, podemos aplicar polimorfismo y distinguir entre clientes físicos y jurídicos.

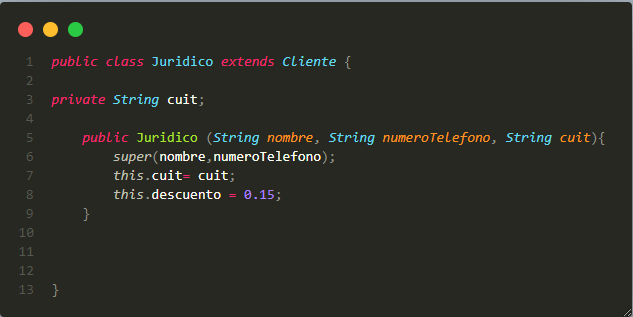
## Refactoring a aplicar: Replace Conditional with Polymorphism./ push down method./ push down fields



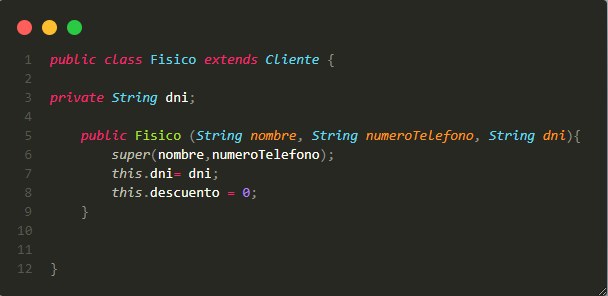
Clase Cliente:



Clase Juridico:

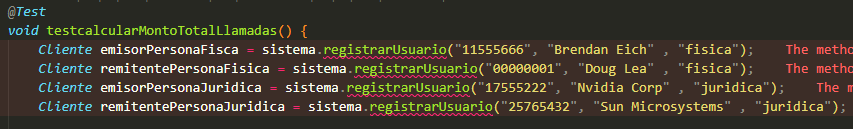


Clase Fisico:

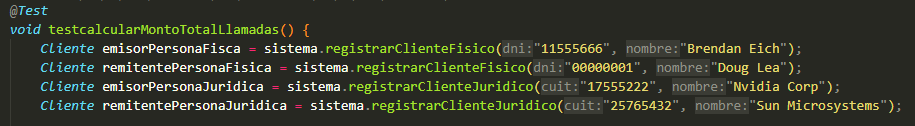


## Esto genera un cambio en el test.

**Antes:**

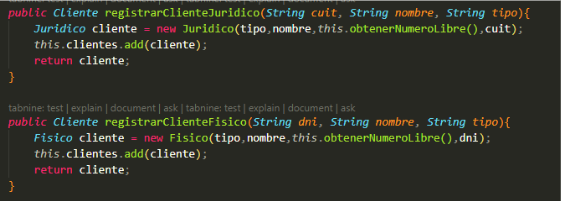


**Despues:**

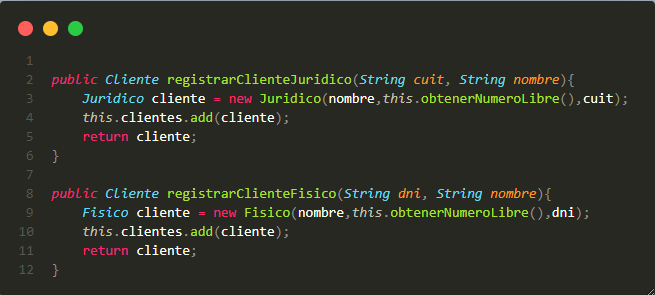


# 2- Mal olor detectado: “dead code”

Podemos observar que el parámetro “tipo” ya no es necesario al haber realizado la jerarquía anteriormente.

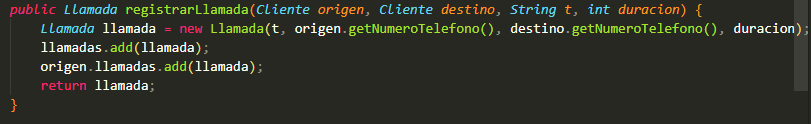


## Refactoring aplicado: “remove parameter”

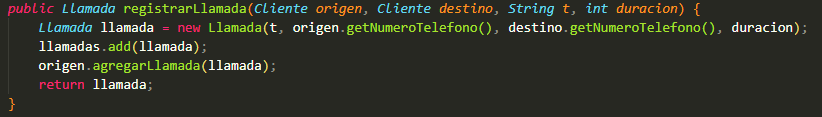


# 3- Mal olor detectado: “Feature envy”

El método registrarLlamada toma un atributo del cliente origen y trabaja sobre él, por lo que se considera envidia de atributos y necesitamos crear el método a la clase cliente para que agregue una llamada.



## Refactoring aplicado: “Extract method”



# 

# 

# 

# 

# 

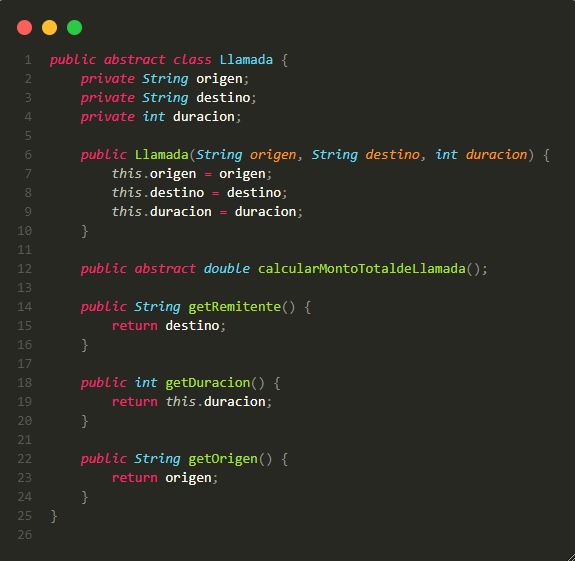
# 4- Mal olor detectado: “Switch statements”

Al tener ifs dentro del método “calcularmontototalLlamadas”, podemos aplicar polimorfismo y distinguir entre llamadas nacionales e internacionales, y entre clientes jurídicos y físicos. De esta forma también se elimina la variable tipodeLlamada.

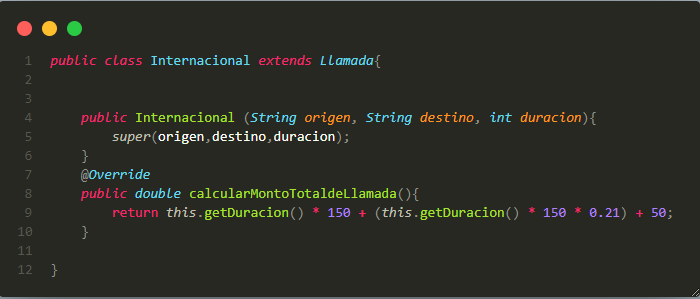


## Refactoring aplicado: Replace Conditional with Polymorphism/ “remove dead variable”

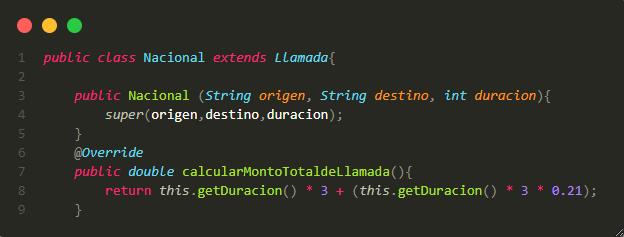
Clase Llamada:



Clase Internacional:

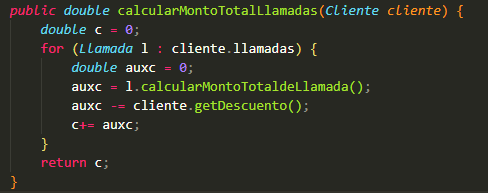


Clase Nacional:



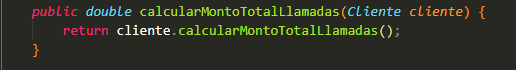
# 5- Mal olor detectado: “Feature envy”

En el método calcularMontoTotalLlamadas podemos observar que se utiliza el atributo de cliente “llamadas” por lo que se considera envidia de atributos y debe moverse el método hacia la clase del propio cliente.

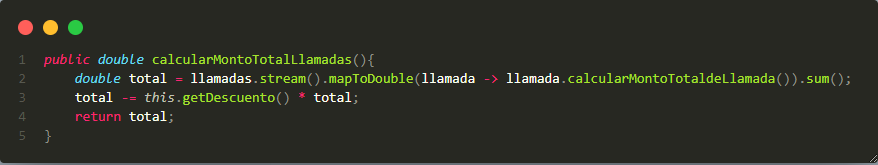


## Refactoring aplicado: “Move method”/ “Push down methods”

Clase Empresa:

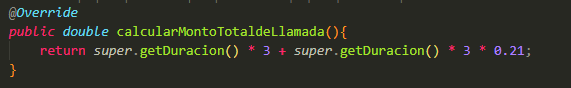


Clase Cliente:

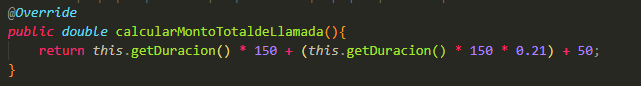


Clase Llamada:



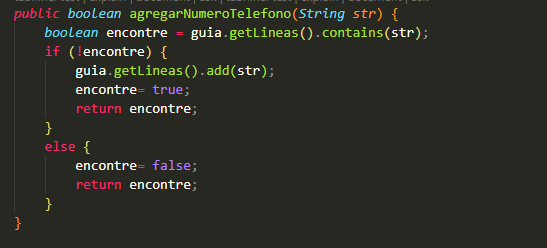
Subclase de llamada, Nacional:  


Subclase de llamada, Internacional:



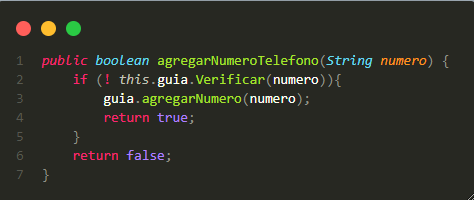
# 6- Mal olor detectado: “Feature envy”

En el método agregarNumeroTelefono podemos observar que se utiliza el atributo de la guia “líneas” por lo que se considera envidia de atributos y debe moverse el método hacia la clase GestorNumerosDisponibles.

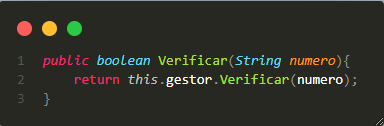


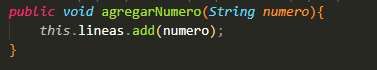
## Refactoring aplicado: “Move method”/ “extract method”

Clase Empresa:



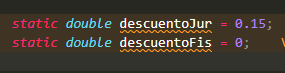
Clase GestorNumerosDisponibles:





# 7- Mal olor detectado: “Dead code”

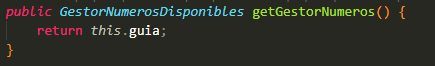
En estas líneas se establecen dos variables estáticas para los distintos descuentos, al utilizar el polimorfismo estos datos son atributos del cliente jurídico y físico respectivamente.



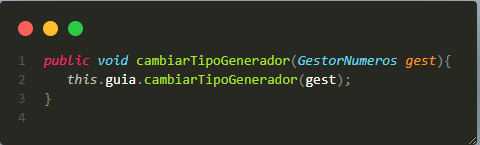
## Refactoring aplicado: “Remove dead code”.

# 8- Mal olor detectado: “Exposed Field”

Este método rompe completamente el encapsulamiento, por lo que se lo encapsula y busca otra manera de acceder a la guia para utilizar los métodos de esta misma.



## Refactoring aplicado: “rename method”/“add parameter”



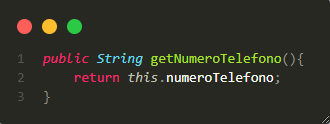
# 9- Mal olor detectado: “Exposed Field”

Al tener un get y set para cada atributo de la clase se rompe el encapsulamiento, por lo que se eliminan los getters y setters que no son necesarios.

Clase Persona:



## Refactoring aplicado: “Remove setting method”.



# 10- Mal olor detectado: “Exposed Field”

Tener un atributo público rompe el encapsulamiento, por lo que se debe cambiar el el alcance.



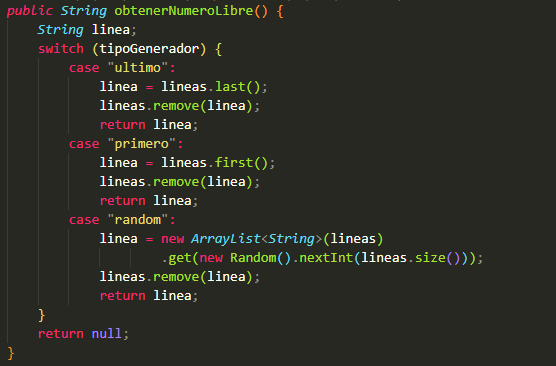
## Refactoring aplicado: “Encapsulate field”



# 11- Mal olor detectado: “Switch statements”

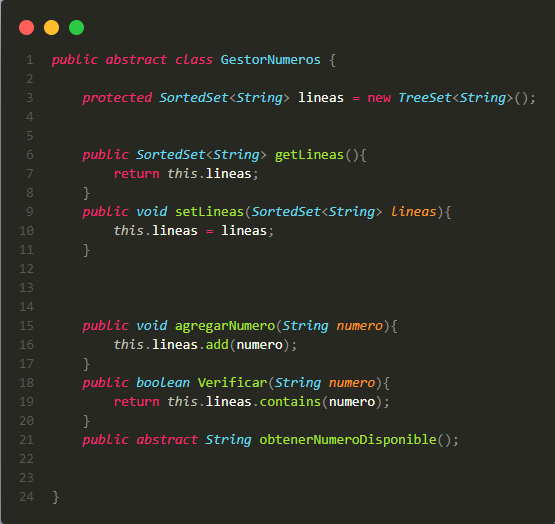
La sentencia case dentro del método “obtenerNumeroLibre” se puede cambiar por polimorfismo distinguiendo entre “PrimerNumero”, “UltimoNumero” y “NumeroRandom”. Esto genera un push down de los métodos a las clases concretas y la creación de metodos get y set para los cambios de variable de instancia. (ej: queremos cambiar el tipo de generador a primer numero de las lineas, entonces pasamos una instancia de “PrimerNumero” y antes de asignarla tenemos que settear a esta instancia con la coleccion de numeros disponibles)

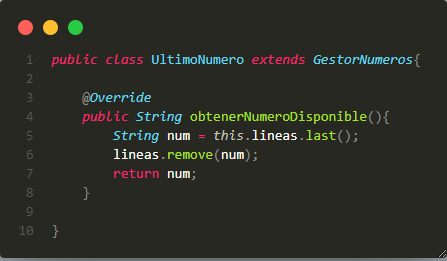
Esto genera cambios en el test.

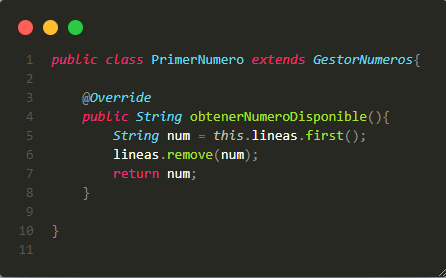


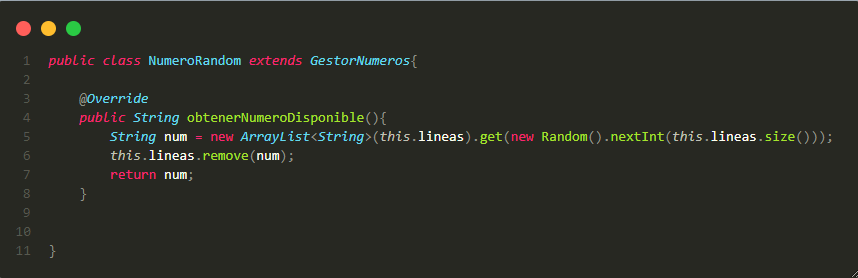
## Refactoring aplicado: “Replace Conditional with Polymorphism”/ “add variable”/ “push down methods”/ “rename method”/ “move field”





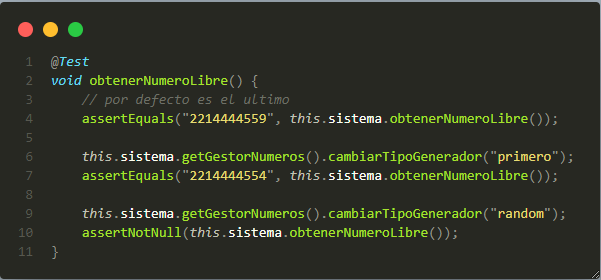




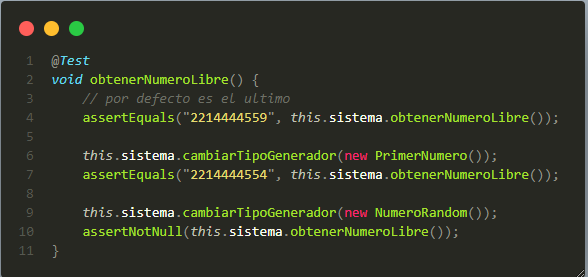


# Cambios en el test:

**Antes:**

****

**Despues:**

****

# 

# 12- Mal olor detectado: “dead code”

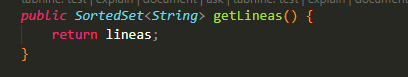
Al realizar la jerarquía anterior, el atributo tipoGenerador ya no es útil, por lo que se elimina.



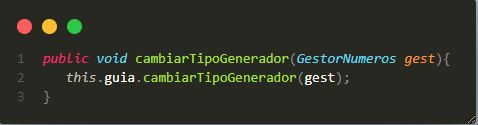
## Refactoring utilizado: “Remove variable”

# 13- Mal olor detectado: “Expose field”

Al igual que en la clase empresa, el getLineas rompe el encapsulamiento. Por lo que agregamos un parámetro para poder acceder y utilizar el metodo cambiarTipoGenerador.



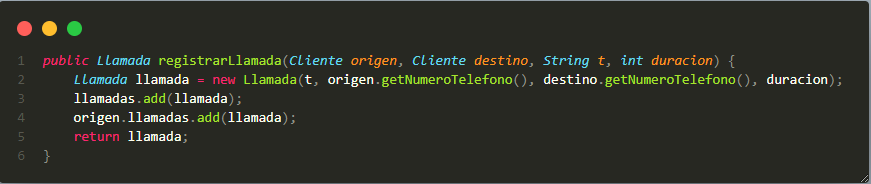
## Refactoring aplicado: “Add Parameter”/ “rename method”



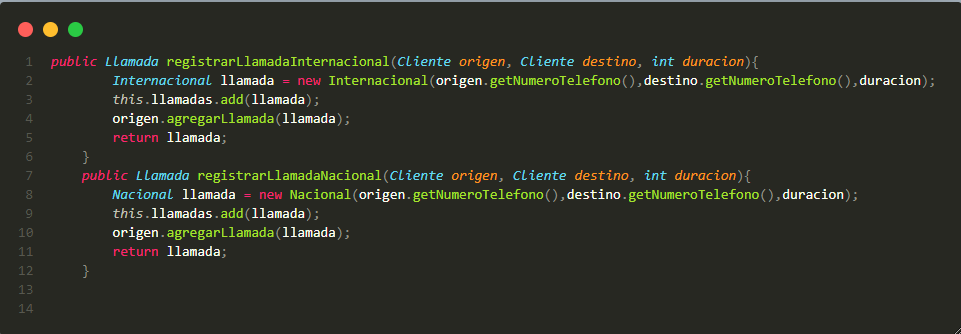
# 

# 14- Mal olor detectado: “Dead code”

En este caso tenemos 4 parámetros de los cuales el “t” que referencia al tipo de llamada ya no nos es útil por la jerarquía antes aplicada.



## Refactoring aplicado: “Remove parameter”

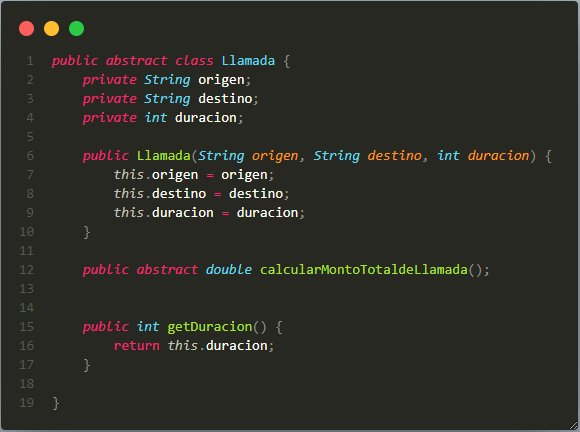


# 15- Mal olor detectado: “Exposed fields”

Los getters como “getRemitente” o “get Origen” no son utilizados, por lo que se los eliminan para no romper el encapsulamiento.



## Refactoring aplicado: “Remove setting method”

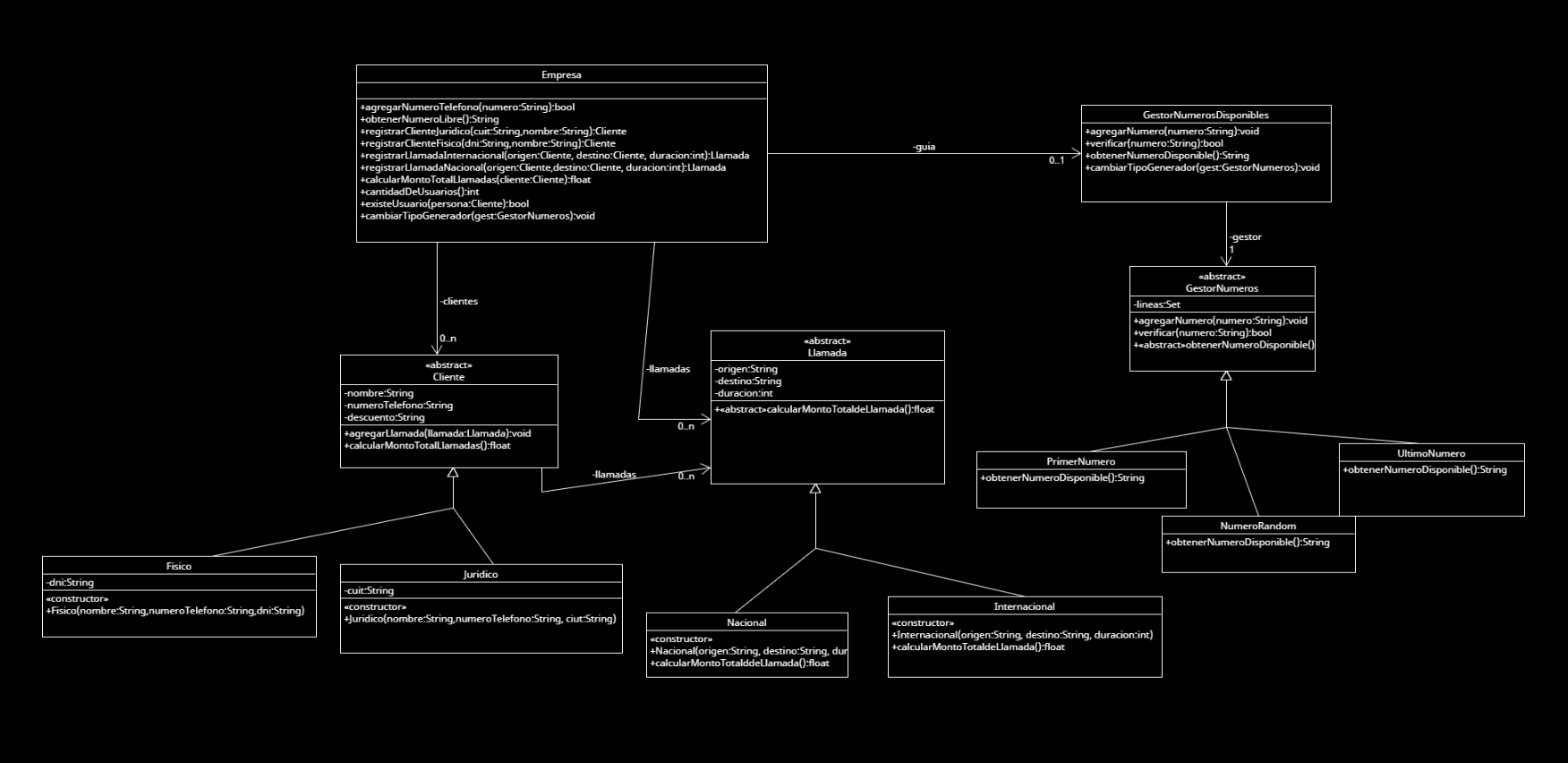


Aclaración: En cuanto al alto acoplamiento que existe entre la clase empresa y Llamada, a nuestra forma de verlo esto puede ser una especificación del dominio del problema y no lo vemos como un mal olor.

# **-UML luego de refactorizar el código:**

En caso que no se pueda ver bien en la foto, dejo el link con el archivo del UML.

<https://drive.google.com/file/d/1wT6_sg99CxO9qMKDoUnUdxhqdQojB3rY/view?usp=sharing>



# **Código completo refactorizado:**

<https://github.com/frnJJ/OO2/tree/main/ej3refact/src/main/java/ar/edu/unlp/info/oo2/facturacion_llamadas>