-Eliott Santistevan

-Leonardo Pincay

-Ronny García

-Melanie Sandoval

Taller 6- Refactoring

INTEGRANTES:

Tabla de contenido

[**Large Class:** 2](#_Toc92392176)

[Tratamiento (Extract class) : 2](#_Toc92392177)

[**Primitive Obsession and Long Parameter:** 2](#_Toc92392178)

[Tratamiento(Replace Data Value with ObjectOriginal text): 3](#_Toc92392179)

[**Feature Envy:** 3](#_Toc92392180)

[Tratamiento(Move Method): 3](#_Toc92392181)

[**Duplicate Code** 4](#_Toc92392182)

[Tratamiento (Substitute Algorithm): 4](#_Toc92392183)

[**Lazy Class:** 4](#_Toc92392184)

[Tratamiento(Inline Class): 4](#_Toc92392185)

[**Duplicate Code** 5](#_Toc92392186)

[Tratamiento (Extract SuperClass): 5](#_Toc92392187)

[**Lazy class:** 7](#_Toc92392188)

[Tratamiento(Inline class): 7](#_Toc92392189)

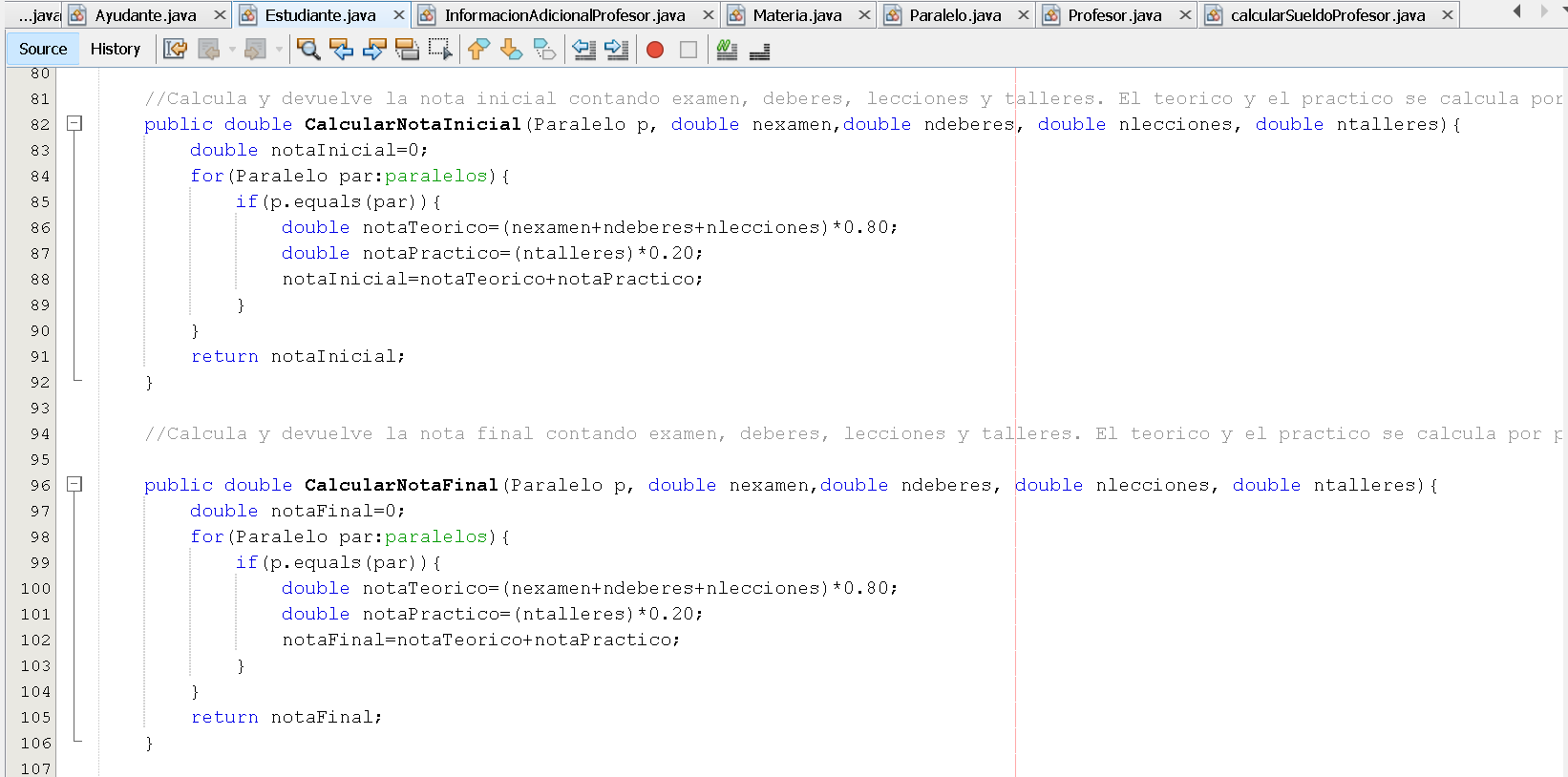
# **Large Class:**

En la clase Estudiante esta tiene métodos que podrían estar implementados en otra clase dando el code smells de clase larga una de las consecuencias de tener esto sería el manejo de dicha clase debido a su complejidad por motivos de que tiene métodos que podrían estar en otras clases.

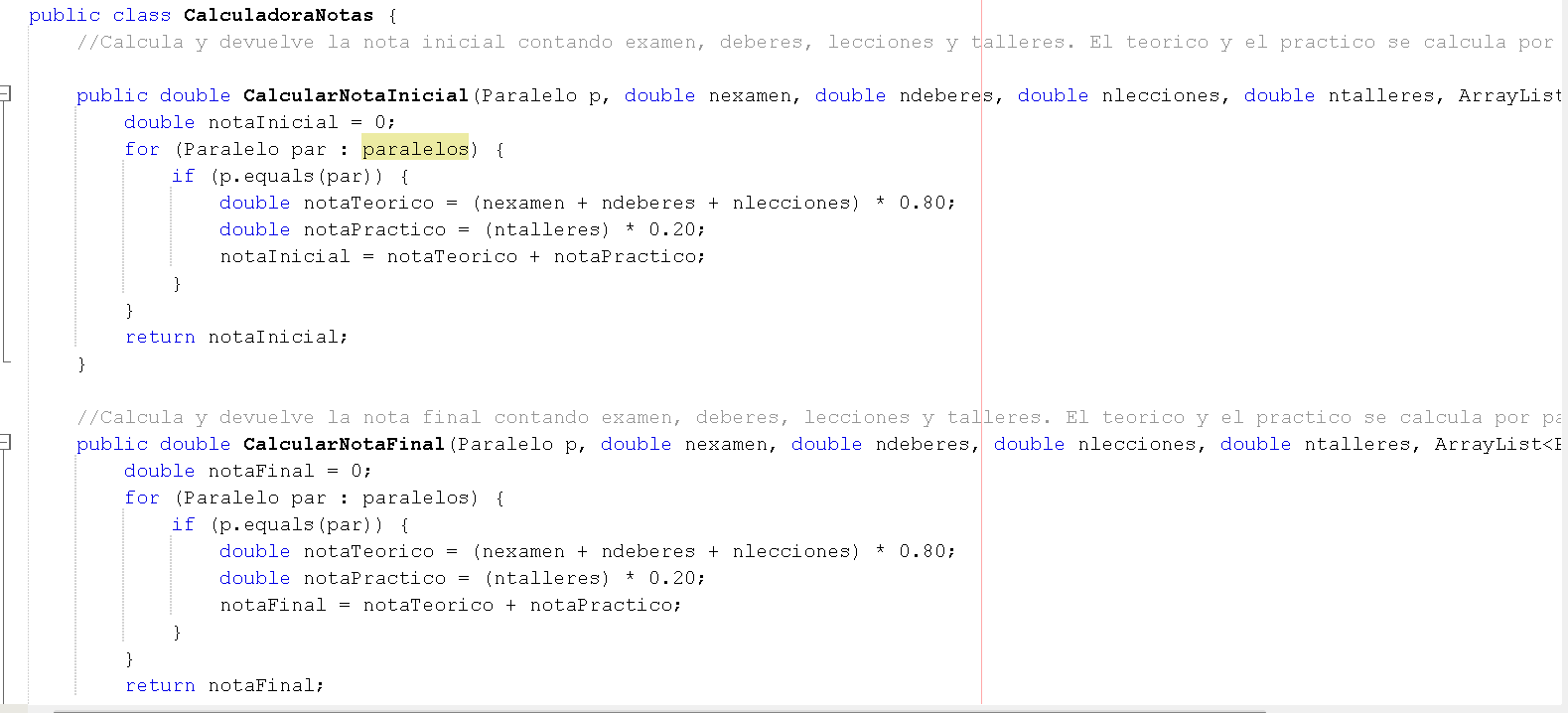
## Tratamiento (Extract class) :

Crear otra clase el cual contenga estos métodos extraídos de la clase Estudiante simplificando esta clase para que su manejo no sea tan complejo como en la version anterior.

**Antes:**

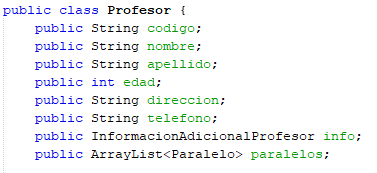
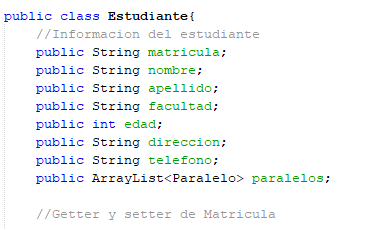


**Después:**



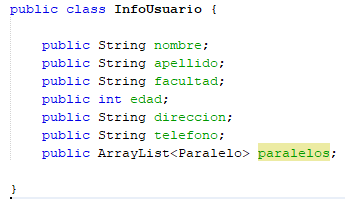
# **Primitive Obsession and Long Parameter:**

En las clases Estudiante y Profesor vemos un uso varios atributos con tipos de datos primitivos, como números de teléfono, códigos, direcciones, etc. A demás esta información hace que existan demasiados parámetros en los constructores de ambas clases.



## Tratamiento(Replace Data Value with ObjectOriginal text):

Creamos una nueva clase datos de usuario en donde pondremos todos los atributos repetidos en ambas clases de tipo String (nombre, apellido, facultad,telefono,etc). Ahora esta información se guarda en el obejto InfoUsuario.

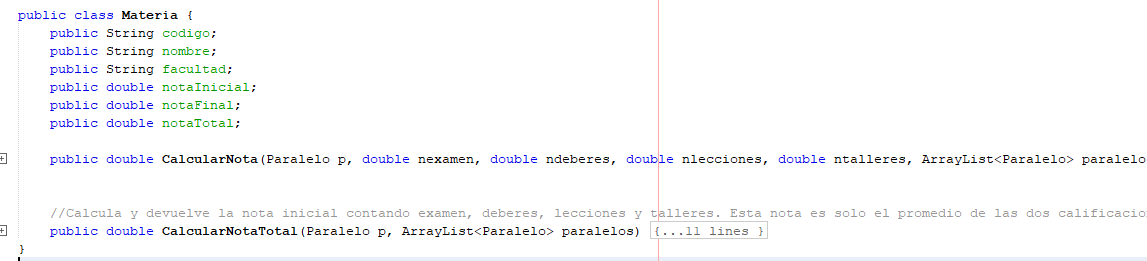


# **Feature Envy:**

En la clase Estudiante tenemos varios métodos como CalcularNotaInicial, CalcularNotaFinal, CalcularNotaTotal que utilizan atributos de la clase Materia.

## Tratamiento(Move Method):

La solución sería mover los métodos a la clase materia.



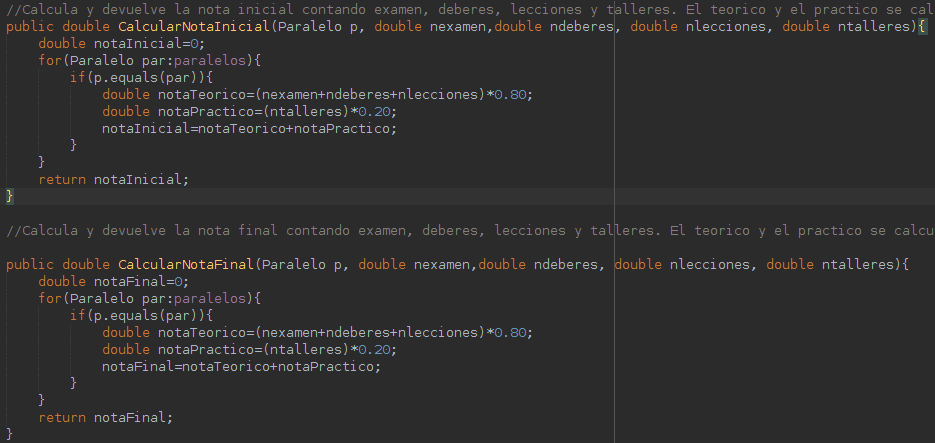
# **Duplicate Code**

Los metodos calcularNotaInicial() y calcularNotaFinal() de la clase estudiante realizan lo misma funcionalidad.

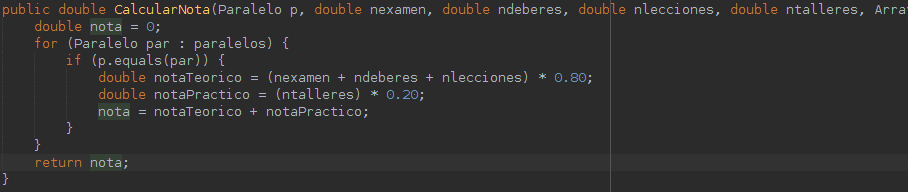
## Tratamiento (Substitute Algorithm):

Se reemplaza los dos algoritmos por uno nuevo más general llamado calcularNota().

**Antes:**



**Después:**



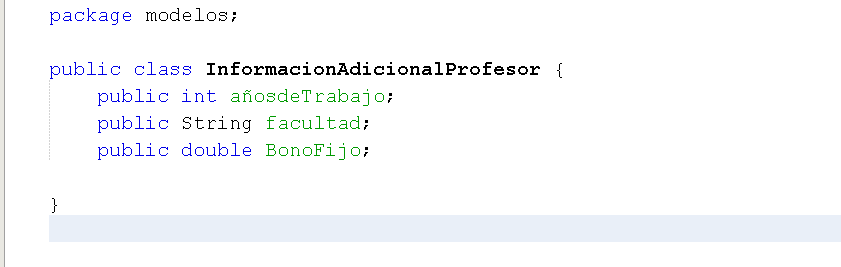
# **Lazy Class:**

La clase InformacionAdicionalProfesor tiene atributos que se le podrían agregar a la misma clase Profesor haciéndola una clase perezosa ya que no cumple con ninguna función adicional que agregarle más atributos a la clase Profesor como consecuencia tenemos una clase que no hace nada haciendo más complejo el sistema y aumentando el número de clases.

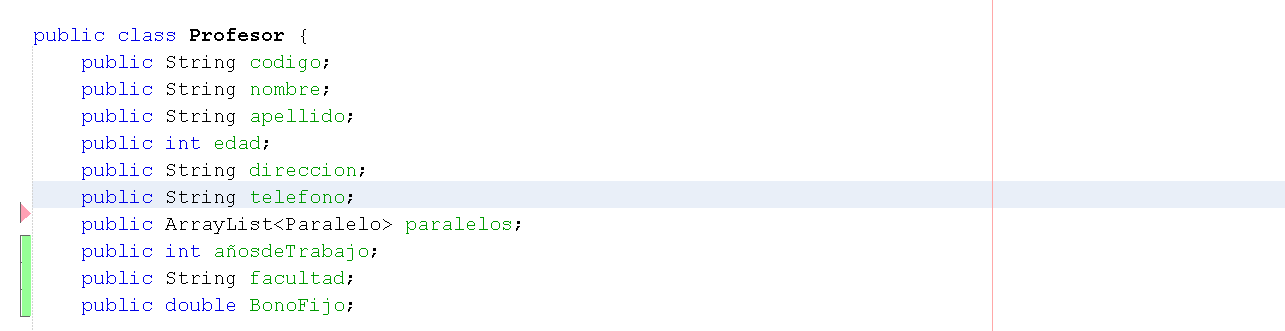
## Tratamiento(Inline Class):

Agrega estos atributos a la clase que más le da uso es decir a la clase Profesor haciendo que esta clase floja sea eliminada, menorando así la carga del sistema con menos clases.

**Antes:**



**Despues:**



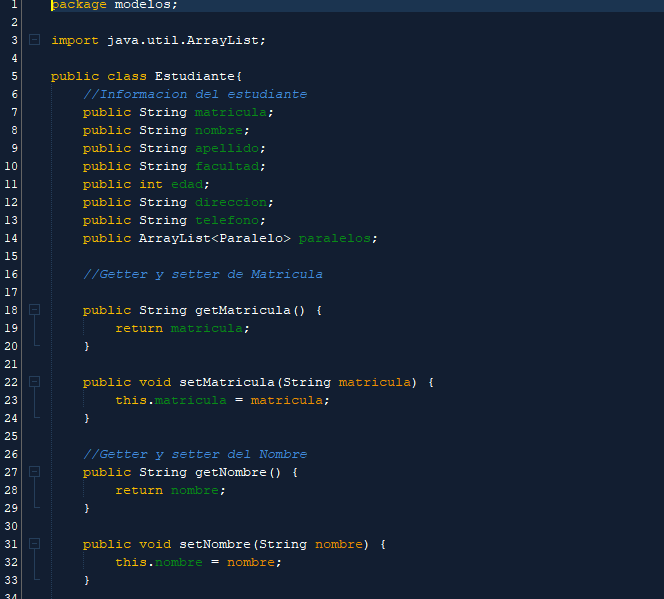
# **Duplicate Code**

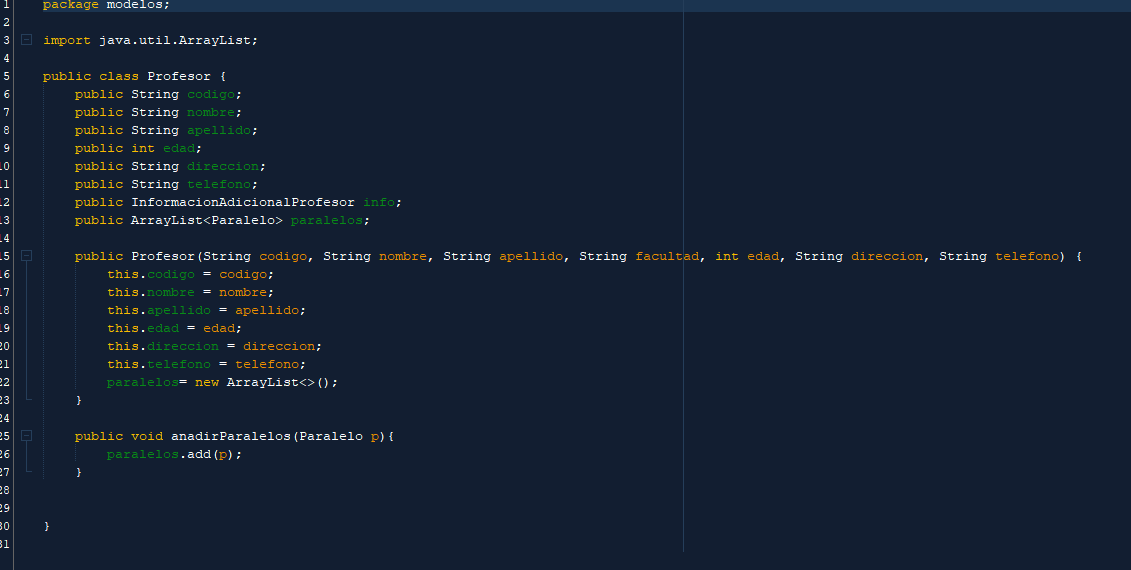
En las clases Estudiante y Profesor existen los mismos atributos, lo que hace que exista código duplicado en ambas clases.

## Tratamiento (Extract SuperClass):

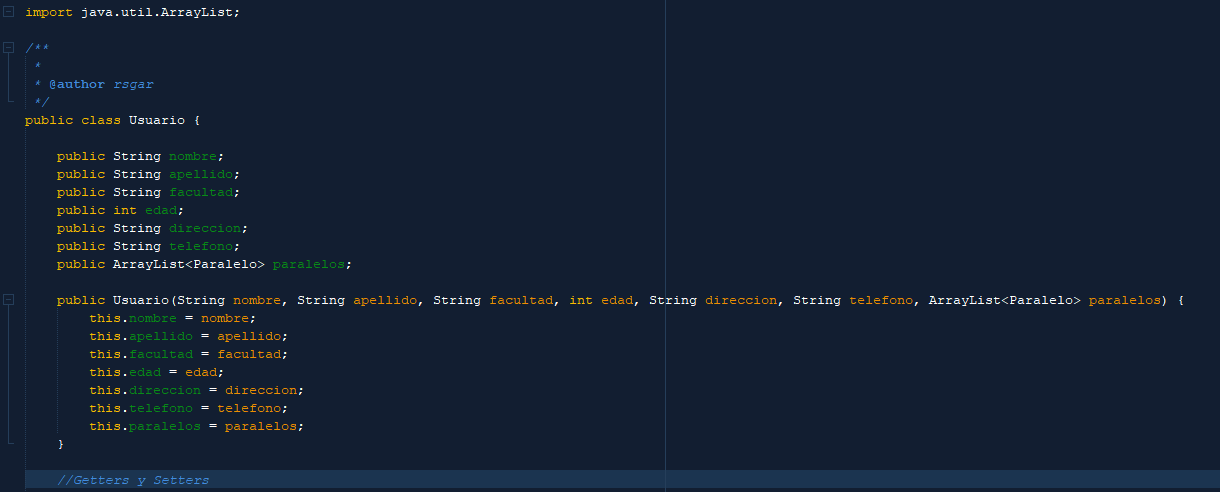
Debemos crear una superclase con todos los atributos de ambas clases con código duplicado, la que en esta ocasión llamaremos “Usuario”, y haremos que las clases Estudiante y Profesor extiendan de la clase Usuario, para así evitar el código duplicado.

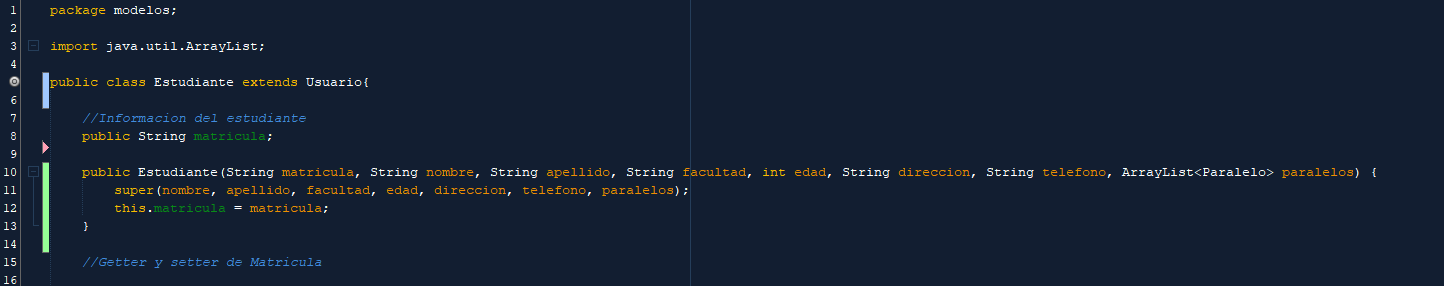
**Antes:**

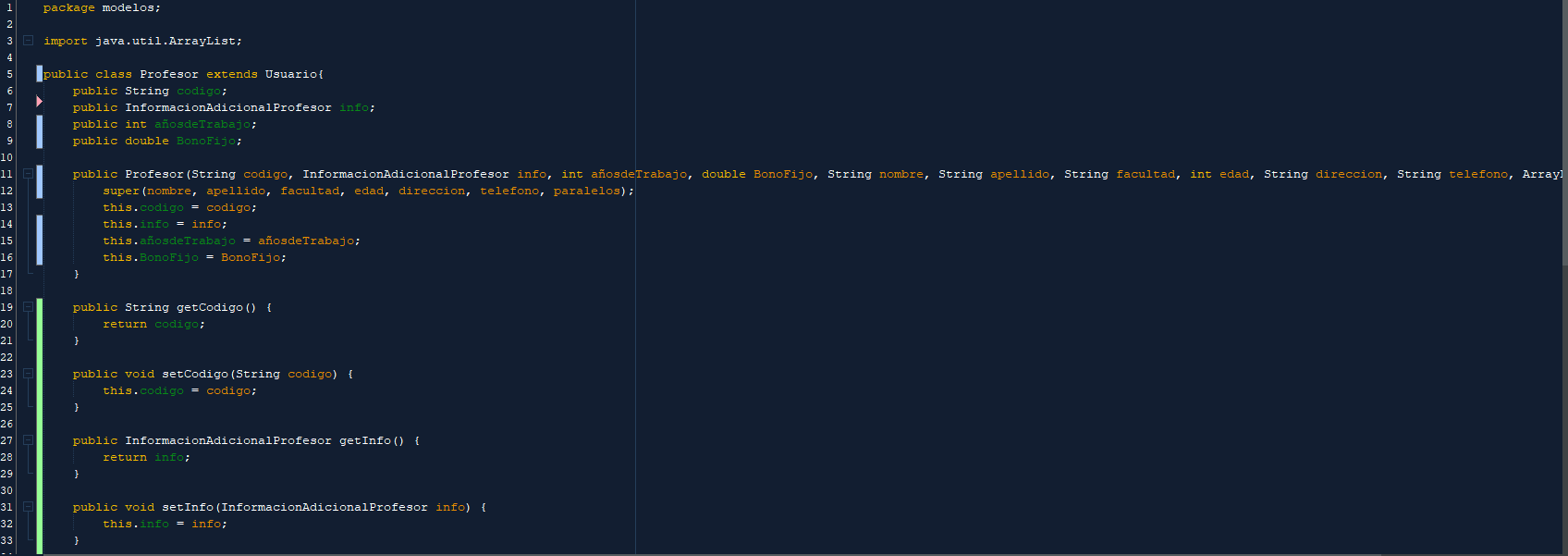




**Después:**







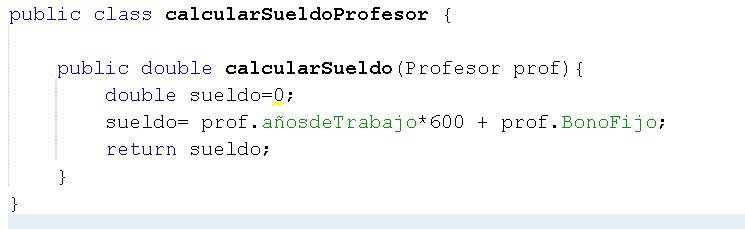
# **Lazy class:**

La clase calcular sueldo profesor es una clase floja debido a que solo contiene un método y no tiene ninguna funcionalidad más y esto daría más complejidad al sistema ya que tendría una clase de más que se podría implementar el método de esta en otra clase que utilice el método

## Tratamiento(Inline class):

Para solucionar esto debemos tomar este método solitario que está en la clase calcularSueldoProfesor e implementarlo en la clase Profesor que es donde se utiliza más este método

**Antes:**



**Después:**

