

Diseño de software

Taller Refactoring

**Equipo 1 - Integrantes:**

* Maria Alexandra Ramos Pluas
* Maria Melina Macias Quinto
* Andres Ismael Toala Buitron
* Dylan Escala Bonilla
* Joel Alejandro Alvarado Jiménez

Enlace de Github utilizado: <https://github.com/DylanEscala/TallerRefactoring>

Contenido

[CODE SMELLS IDENTIFICADOS: 3](#_Toc59652699)

[Duplicate code. 3](#_Toc59652700)

[ Clase Estudiante 3](#_Toc59652701)

[Long Parameter list. 5](#_Toc59652702)

[ Clase Estudiante 5](#_Toc59652703)

[Lazy Class. 6](#_Toc59652704)

[ Clase InfoAdicionalProfesor 6](#_Toc59652705)

[Temporary Field 6](#_Toc59652706)

[· Clase CalcularSueldoProfesor 6](#_Toc59652707)

[Data Class 7](#_Toc59652708)

[· Clase Materia 7](#_Toc59652709)

[Speculative Generality 9](#_Toc59652710)

[· Clase Paralelo 9](#_Toc59652711)

[Replace Delegation with Inheritance 10](#_Toc59652712)

[· Clase Ayudante 10](#_Toc59652713)

[Comments 11](#_Toc59652714)

[· Clase Estudiante y otras clases más. 11](#_Toc59652715)

# CODE SMELLS IDENTIFICADOS:

## Duplicate code.

### Clase Estudiante

#### *Consecuencias.*

En la clase **Estudiante**, algunas variables de instancia y sus métodos getters/setters se repiten en la Clase **Profesor** lo cual genera código duplicado entre dos clases. Además, en los métodos **CalcularNotaInicial y CalcularNotaFinal** de la clase **Estudiante** tienen el mismo algoritmo, lo que causa un aumento de líneas innecesarias y código duplicado.

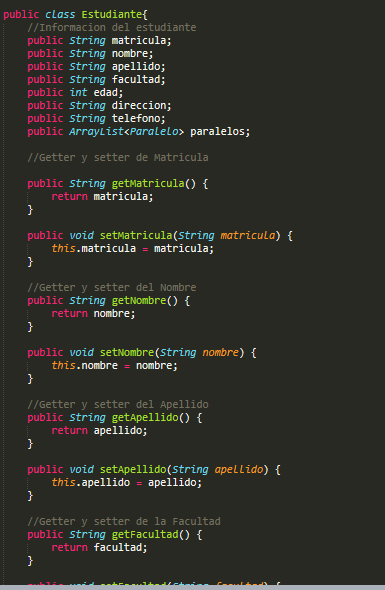
#### *Refactoring.*

Aplicamos **Extract Superclass (Pull up field, method and Constructor body)** en código duplicado en dos clases distintas y de esta forma mantenemos una superclase denominada **Persona** entre estas dos clases y mantener su funcionabilidad.

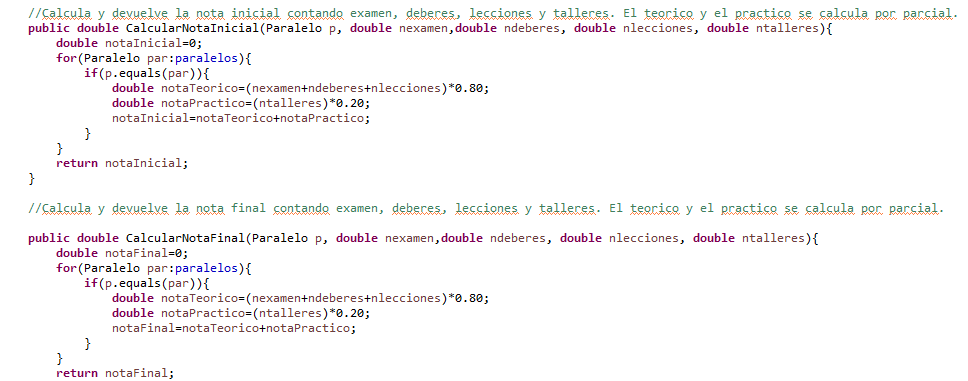
Aplicamos **Extract Method** en código duplicado en dos o más métodos y de esta forma usar un solo método llamar **CalcularNota** y cuando lo usemos, ya dependerá del cliente si es una nota inicial o final.

#### Captura Inicial.

##### Duplicate code encontrado en dos clases distintas (profesor y estudiante).

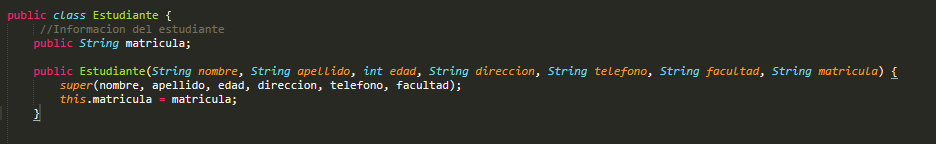


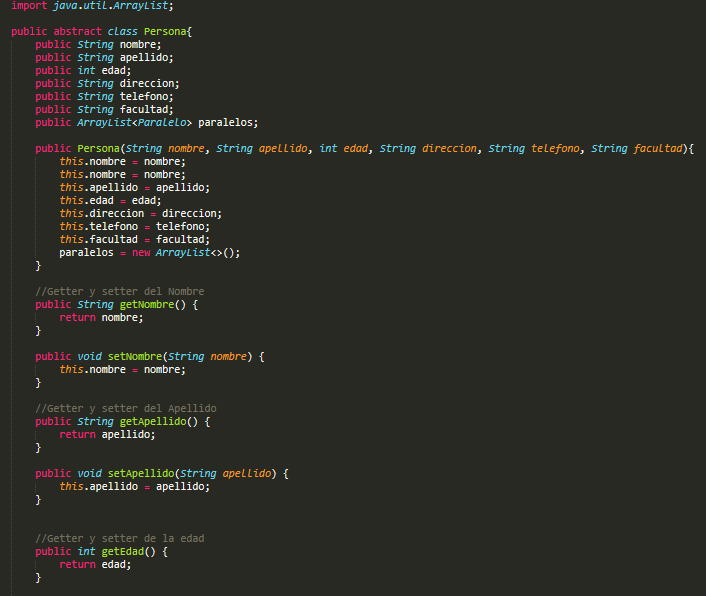
##### Duplicate code encontrado en dos métodos.



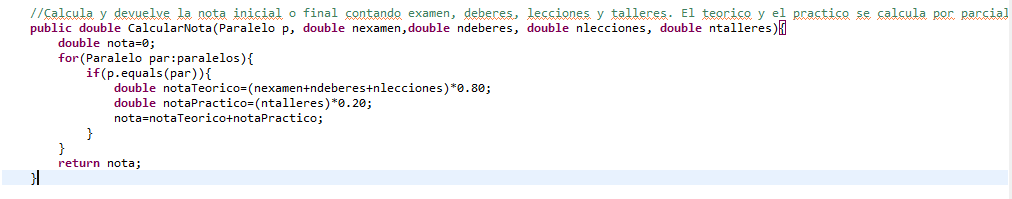
#### Captura código refactorizado:

##### Duplicate code encontrado en dos clases distintas refactorizado(profesor y estudiante).





##### Duplicate code encontrado en dos métodos refactorizado.



## Long Parameter list.

### Clase Estudiante

Dentro de esta clase se puede apreciar el método **“CalcularNota”** que posee más de tres parámetros para realizar el algoritmo correspondiente.

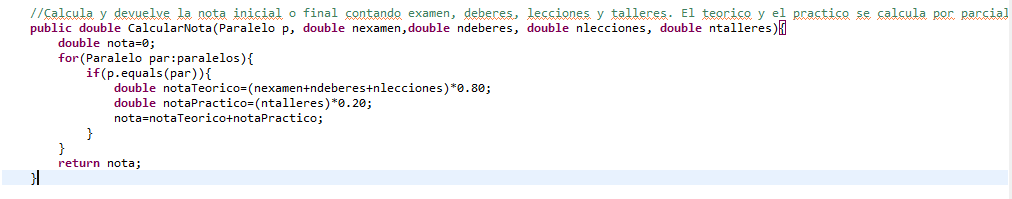
#### Consecuencias.

* + - * Si este mal olor se mantiene, se es propenso a tener un código poco entendible y de gran extensión según el empleo que se les den a los parámetros.
      * Si no se realiza la refactorización existe la posibilidad que existan códigos duplicados no identificados.

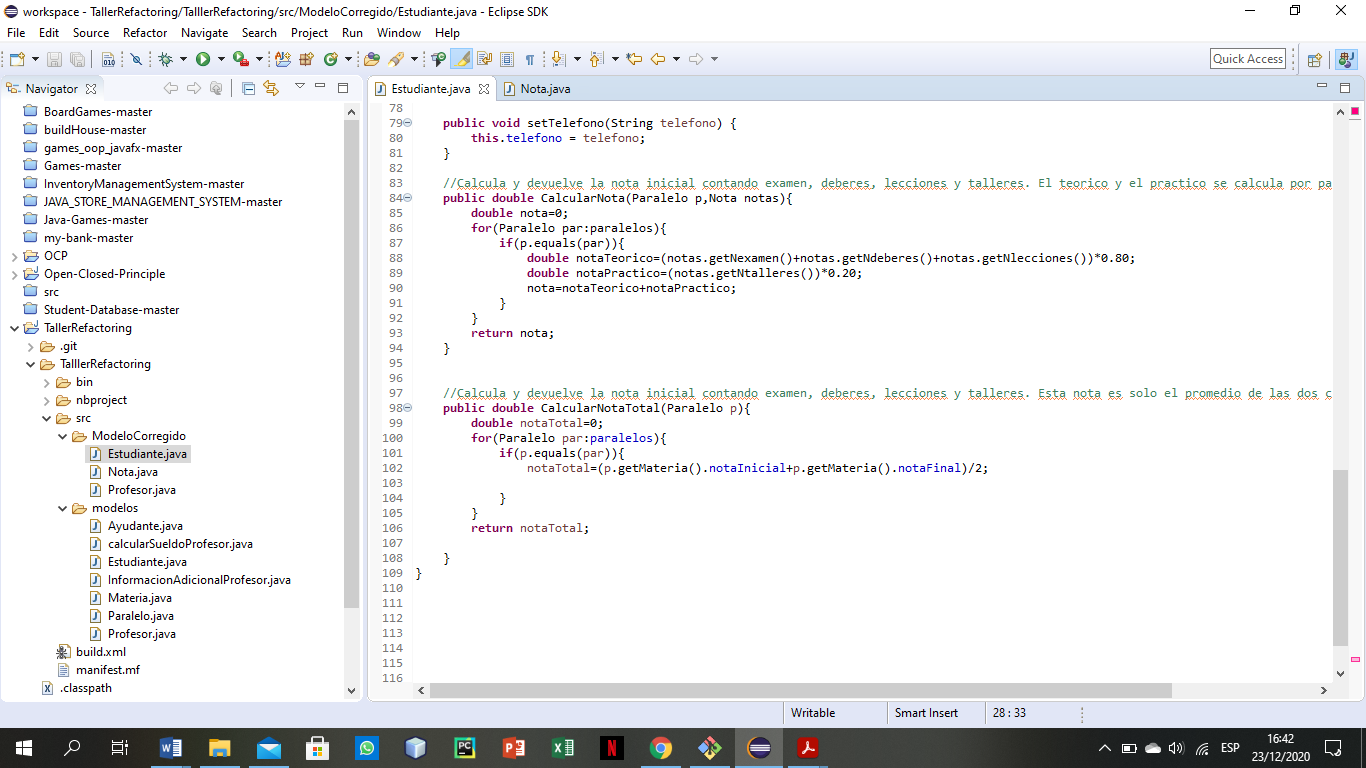
#### Refactoring.

* + - * Se empleo la técnica “[**Introduce Parameter Object**](https://refactoring.guru/introduce-parameter-object)" que separa los datos no relacionados y los combina en un solo objeto que en este caso será **“Nota”**

#### Captura Inicial:



#### Captura Código refactorizado:



## Lazy Class.

### Clase InfoAdicionalProfesor

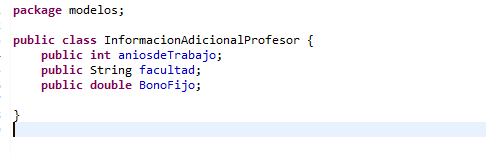
#### Consecuencias.

Tenemos una clase vaga que solo tiene atributos y ocupan espacio de mas por lo que no son usados bajo ningún método dentro de la misma.

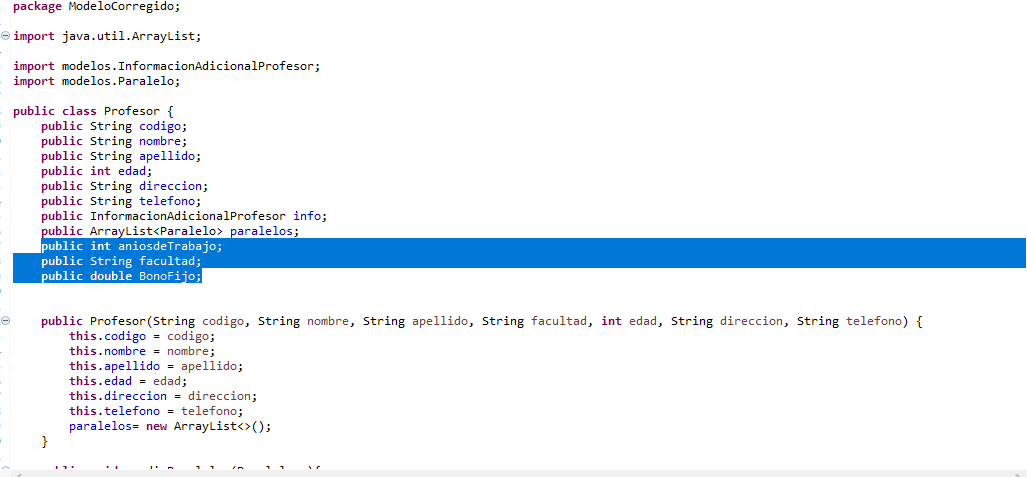
#### Refactoring.

Se aplicará **Collapse Hierarchy** moviendo los atributos de la clase **InfoAdicionalProfesor** a la correspondiente clase **Profesor**.

#### Captura Inicial.



#### Captura Código refactorizado:



## Temporary Field

### Clase CalcularSueldoProfesor

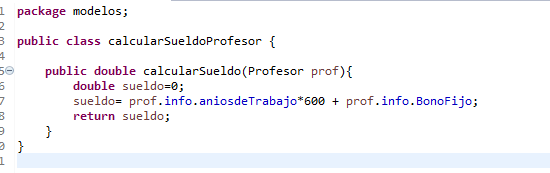
#### Consecuencias.

La variable sueldo se usa solo para asignarle el valor del sueldo calculado y retornarla, pero no aporta en nada, a la lógica del cálculo, y bajo cualquier otra circunstancia tiene valor nulo.

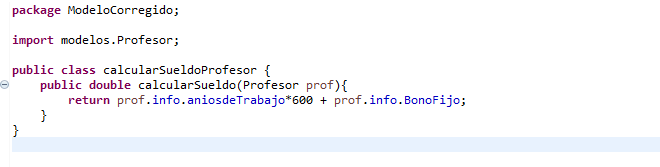
#### Refactoring.

Aplicamos Inline TEMP para retornar directamente el sueldo calculado sin la necesidad de crear una variable adicional innecesaria.

#### Captura inicial:



#### Captura código refactorizado:



## Data Class

### Clase Materia

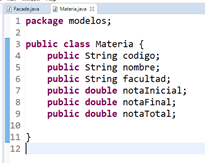
#### Consecuencias.

Ya que estas clases son simples contenedores de datos para otras clases, no puede contener ninguna funcionalidad adicional u operar de manera independiente a las clases que las usan.

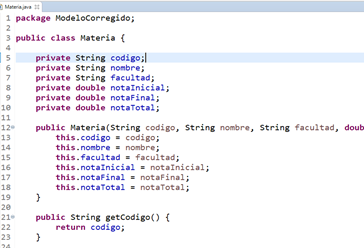
#### Refactoring.

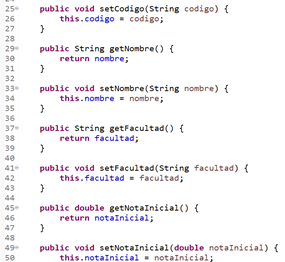
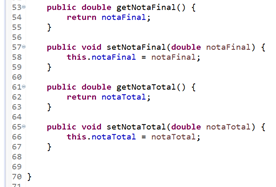
Primero se encapsularán todos los campos públicos con el **Encapsulate Field** y luego se procederá a crear su acceso por medio de getters y setters.

#### Captura inicial.



#### Captura código refactorizado:



## Speculative Generality

### Clase Paralelo

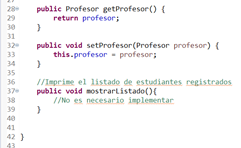
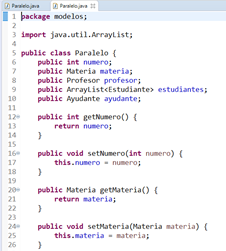
#### Consecuencias.

El método **mostrarListado()** fue creado, pero nunca implementado, quizás para admitir futuras funcionalidades. Como resultado, el código puede volverse difícil de entender y soportar.

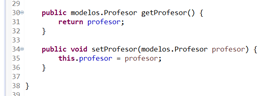
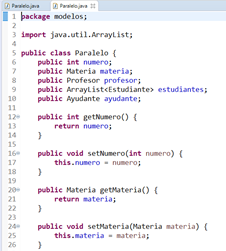
#### Refactoring.

Ya que le método no fue implementado y no recibe parámetros la solución más simple seria eliminarlo por completo.

#### Captura inicial.



#### Captura código refactorizado.



## Replace Delegation with Inheritance

### Clase Ayudante

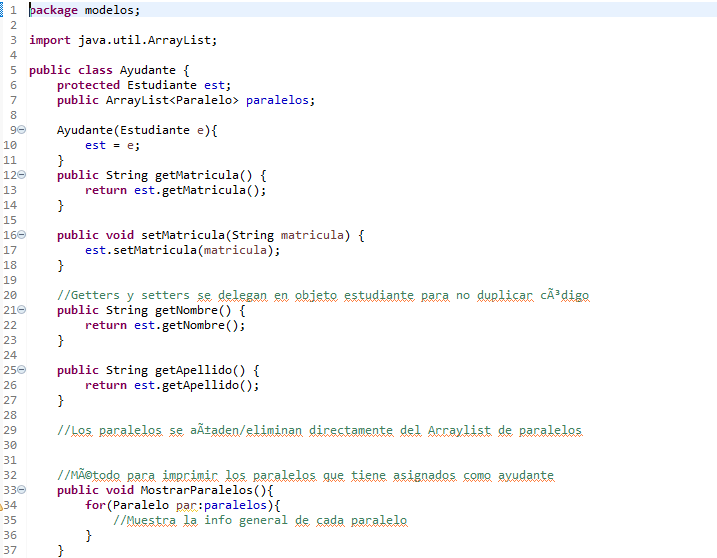
#### Consecuencias.

La clase **Ayudante** contiene como **Estudiante** a atributo, a pesar de que un ayudante es un tipo de **Estudiante**.

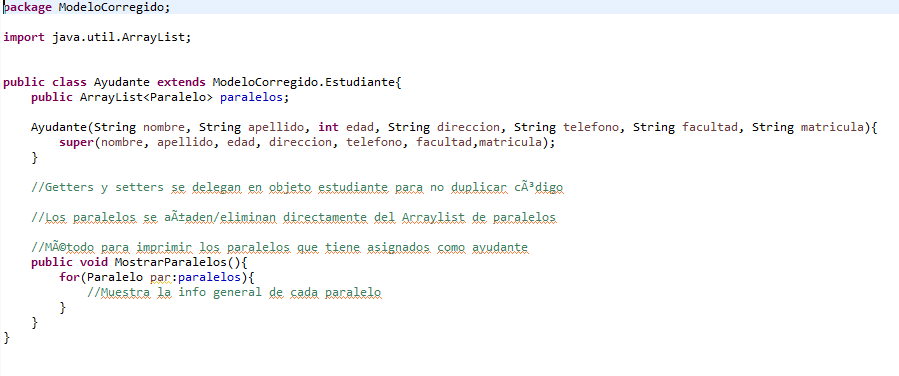
#### Refactoring.

La solución es convertir a **Ayudante** en una clase hija de **Estudiante** para que este herede sus atributos y no necesite de un atributo **Estudiante**

#### Captura inicial.



#### Captura código refactorizado:



## Comments

### Clase Estudiante y otras clases más.

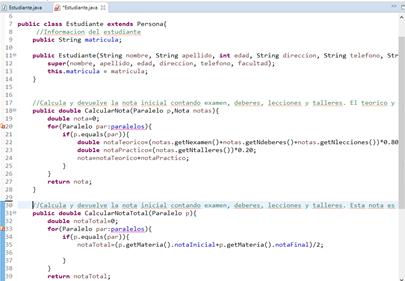
#### Consecuencias.

Los comentarios se usan cuando un fragmento de código no se puede entender por sí solo, por lo que se busca explicarlo por medio de estos.

#### Refactoring.

Como los nombres de los métodos son claros y por medio de estos podemos saber cuál es la finalidad de los métodos, los comentarios se borran.

#### Captura inicial.



#### Captura código refactorizado.

