**A close up of a sign

Description automatically generatedESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**DISEÑO DE SOFTWARE**

**TALLER DE REFACTORIZACIÓN**

**GRUPO 9**

**PARALELO 2**

**INTEGRANTES:**

* ANGIE ARGUDO
* ADRIANA GUILINDRO
* MANUEL LOOR

[Code Smell- Lazy Class. 4](#_Toc48251926)

[Descripción. 4](#_Toc48251927)

[Consecuencias. 4](#_Toc48251928)

[Técnica de Refactorización. 4](#_Toc48251929)

[Código inicial. 4](#_Toc48251930)

[Código refactorizado. 4](#_Toc48251931)

[Code Smell- Duplicate Code. 5](#_Toc48251932)

[Descripción. 5](#_Toc48251933)

[Consecuencias. 5](#_Toc48251934)

[Técnica de Refactorización. 5](#_Toc48251935)

[Código inicial. 5](#_Toc48251936)

[Código refactorizado. 5](#_Toc48251937)

[Code Smell – Data Class. 6](#_Toc48251938)

[Descripción. 6](#_Toc48251939)

[Consecuencias. 6](#_Toc48251940)

[Técnica de refactorización. 6](#_Toc48251941)

[Código Inicial. 6](#_Toc48251942)

[Código refactorizado. 6](#_Toc48251943)

[Code smell – Dead Code. 7](#_Toc48251944)

[Descripción. 7](#_Toc48251945)

[Consecuencias. 7](#_Toc48251946)

[Técnica de refactorización. 7](#_Toc48251947)

[Código Inicial. 7](#_Toc48251948)

[Código refactorizado. 7](#_Toc48251949)

[Code smell – Message Chains. 8](#_Toc48251950)

[Descripción. 8](#_Toc48251951)

[Consecuencias. 8](#_Toc48251952)

[Técnica de refactorización. 8](#_Toc48251953)

[Código inicial. 8](#_Toc48251954)

[Código refactorizado. 8](#_Toc48251955)

[Code smell – Innappropriate Intimacy. 9](#_Toc48251956)

[Descripción. 9](#_Toc48251957)

[Consecuencias. 9](#_Toc48251958)

[Técnica de refactorización. 9](#_Toc48251959)

[Código inicial. 9](#_Toc48251960)

[9](#_Toc48251961)

[Código refactorizado. 10](#_Toc48251962)

Code Smell- Lazy Class.

Descripción.

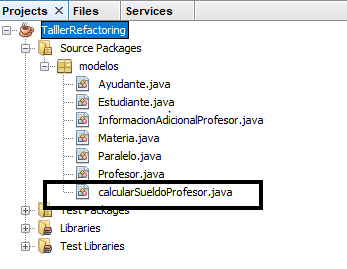
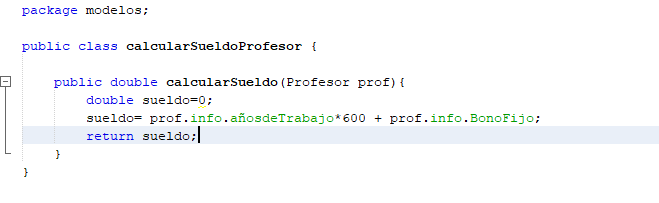
La clase **CalcularSueldoProfesor** es una clase que no tiene ninguna funcionalidad en la implementación del sistema del académico, ya que este solo requiere de consulta de notas y asignación de profesores y ayudantes en un paralelo, por lo que el uso de esta clase no es requerido.

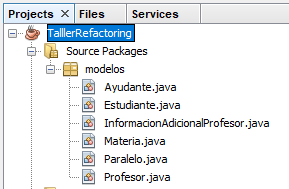
Consecuencias.

Al tener una clase que no tiene ninguna utilidad, esto genera un costo de mantenimiento del sistema innecesario y de tiempo que se podría utilizar para otras clases o mantener el sistema mucho más eficiente y eficaz para los usuarios que utilizan este servicio.

Técnica de Refactorización.

Para las clases que no son utilizadas en el sistema, la forma más sencilla de arreglar este code smell es eliminando la clase, en este caso **CalcularSueldoProfesor**.

Código inicial.

Código refactorizado.

Code Smell- Duplicate Code.

Descripción.

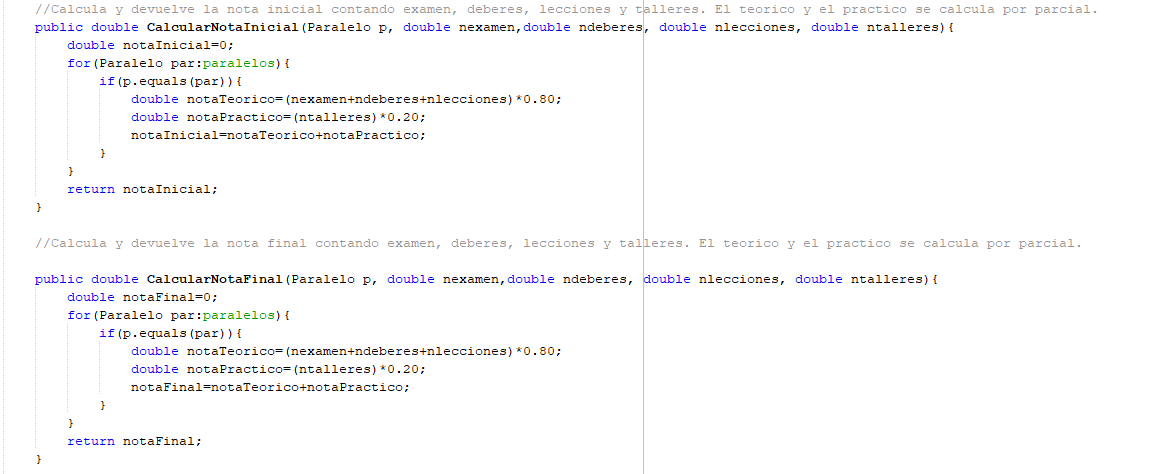
La clase Estudiante posee dos métodos llamados **CalcularNotaInicial()** y **CalcularNotaFinal()** que reciben los mismos parámetros y tienen la misma implementación para calcular la nota del estudiante.

Consecuencias.

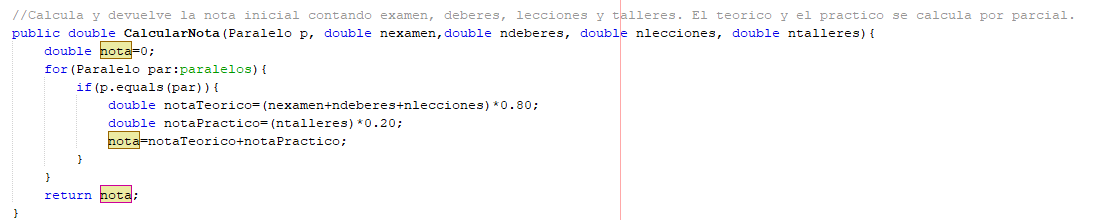
Tener métodos con código repetido entorpece la lectura del código, volviéndolo más largo de lo que debería y también podría acarrear problemas a los usuarios que pueden llegar a confundir si existe alguna diferencia entre el cálculo de ambas notas.

Técnica de Refactorización.

Para la refactorización y corregir este code smell, se puede crear un método general llamado **CalcularNota()** el cual el usuario puede utilizar independientemente de qué nota esté calculando, si es una nota teórica con los trabajos que tiene o la final con todos los datos que posee.

Código inicial.

Código refactorizado.



Code Smell – Data Class.

Descripción.

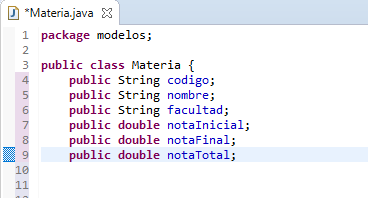
La clase **Materia** emite un code smell de tipo **data class** debido a que el único motivo para acceder a ella es para utilizar sus campos.

Consecuencias.

Mantener esa clase provocará que sea más difícil la organización y comprensión del código, ya que puede existir algún método que requiera de los campos de esa clase. Adicionalmente, será un obstáculo al momento de detectar la duplicación de código relacionado con la clase **Profesor**.

Técnica de refactorización.

La forma más sencilla para corregir el code smell presente en esta clase es aplicando el método **Encapsulate Field**, el cual propone convertir los modificadores de acceso de los campos de la clase a modificadores de acceso privados y crear métodos de acceso para estos campos.

Código Inicial.

Código refactorizado.

Code smell – Dead Code.

Descripción.

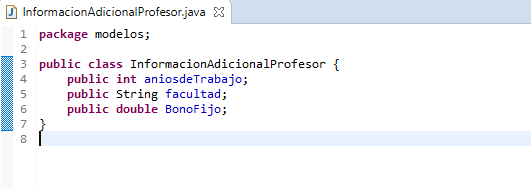
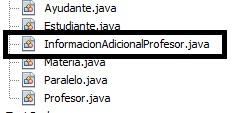
Debido a que se eliminó la clase **CalcularSueldoProfesor**, la clase InformacionAdicionalProfesorya no tiene ningún motivo de existir en nuestro proyecto, por loquela clase es redundante.

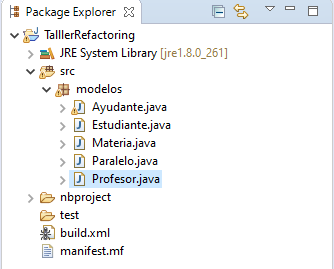
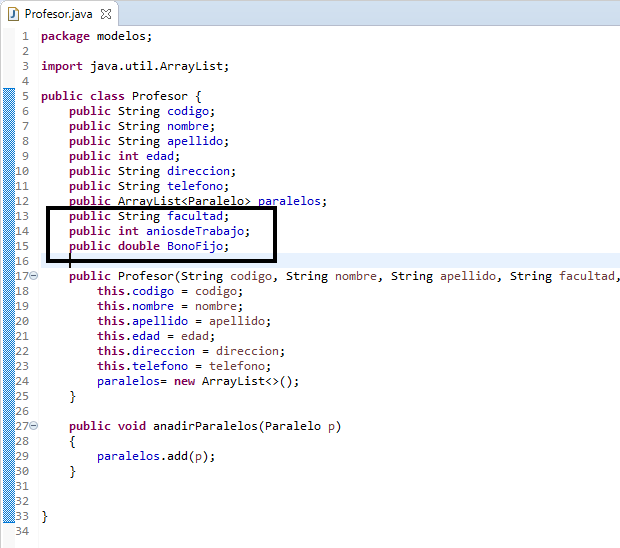
Consecuencias.

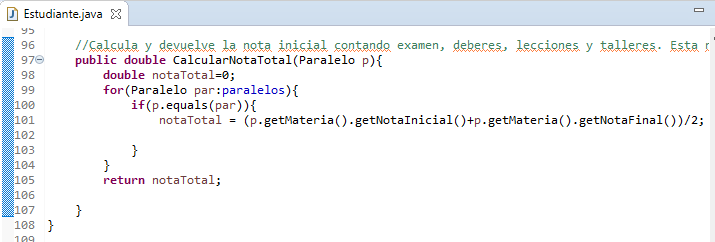
Si mantenemos esta clase, el soporte del código será complejo. Además, desaprovecharíamos la oportunidad de reducir el código.

Técnica de refactorización.

Debido a la actualización del proyecto, la cual fue eliminar la clase **CalcularSueldoProfesor**, la clase **InformacionAdicionalProfesor** es redundante, por lo que es necesario eliminar la clase. Para rescatar los campos de esta clase podemos utilizar el método **Inline Class** y ubicarlos en la clase Profesor.

Código Inicial.

Código refactorizado.

Una consecuencia de haber utilizado el método **Inline Class** es la modificación del método **CalcularNotaTotal** de la clase **Estudiante**:

Code smell – Message Chains.

Descripción.

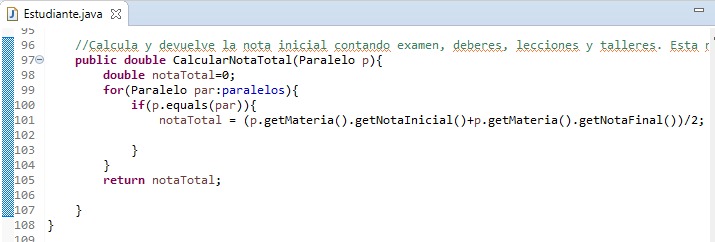
Dentro de la clase Estudiante, se encuentra el método **CalcularNotaTotal()** en el cual se utiliza un objeto de tipo Paralelo para obtener su materia y luego, se utiliza el objeto de Materia para obtener la nota inicial. Lo mismo se realiza para obtener la nota final.

Consecuencias.

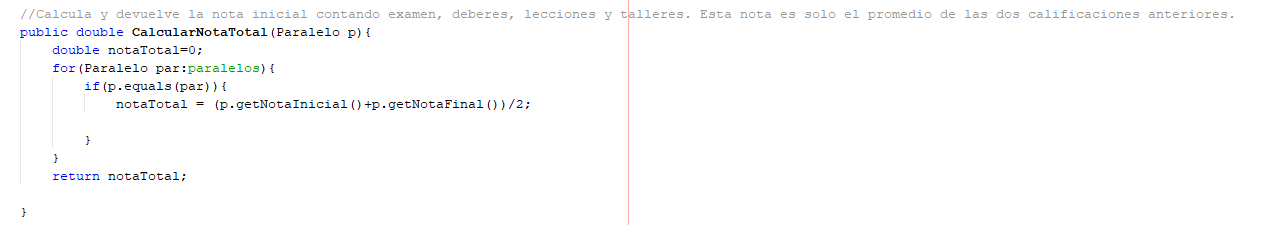
Demuestra dependencia dentro de la estructura de clases y trae complicaciones en futuras modificaciones, lo que hace que el código no sea eficiente.

Técnica de refactorización.

Utilizaremos el método **Hide delegate** creando dos métodos nuevos dentro de la clase Paralelo para obtener la nota inicial y final en dicha materia y de esta forma no se forme una cadena.

Código inicial.

Código refactorizado.



Code smell – Innappropriate Intimacy.

Descripción.

Las clases Ayudante, Estudiante, Paralelo y profesor tienen atributos declarados como públicos, lo que no restringe la relación entre las clases.

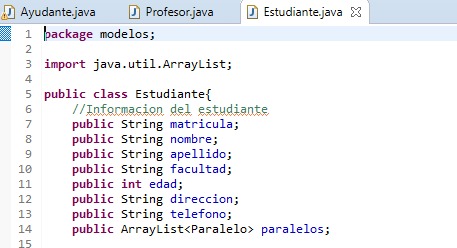
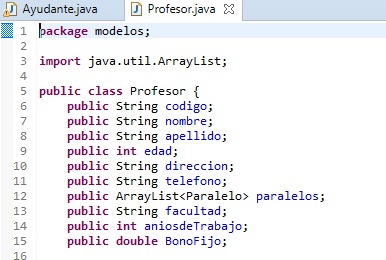
Consecuencias.

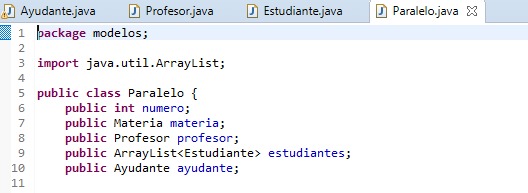
Las clases se encuentran altamente relacionadas cuando deberían conocerse lo menos posible, esto provoca que la calidad del código sea mala.

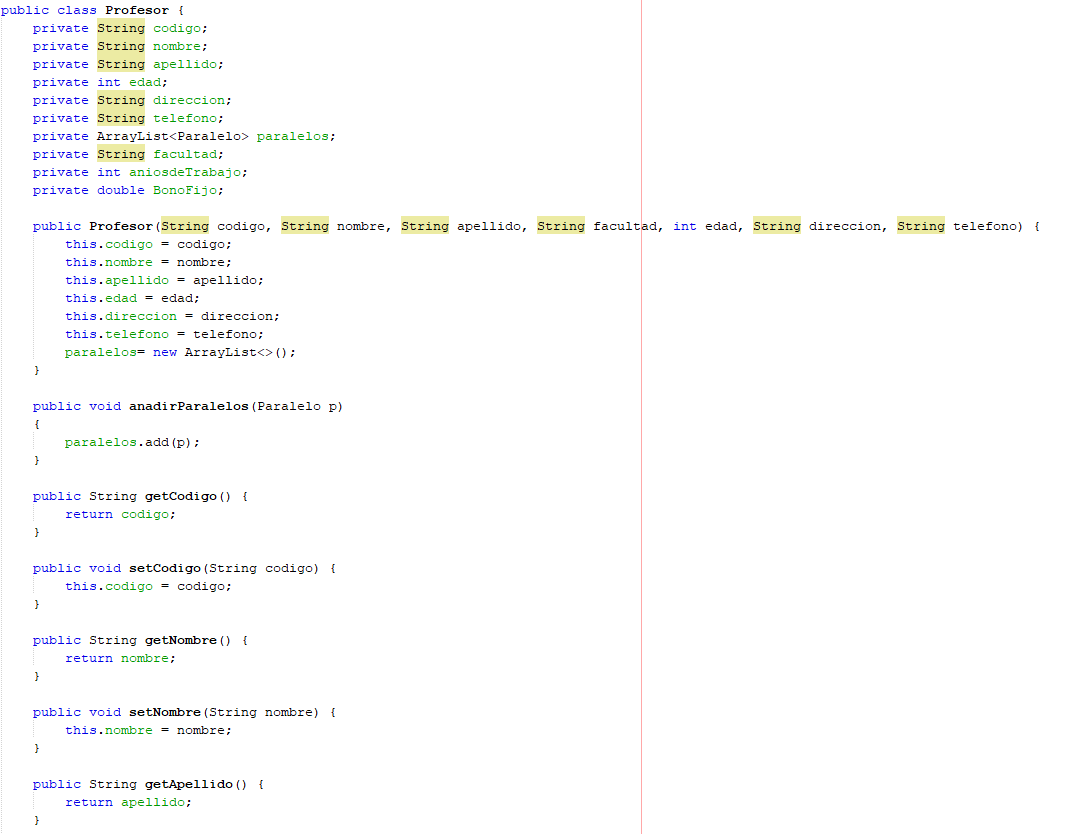
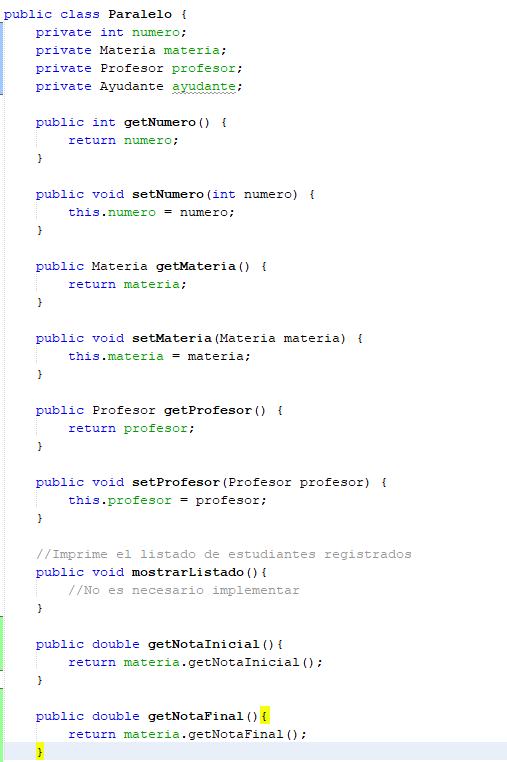
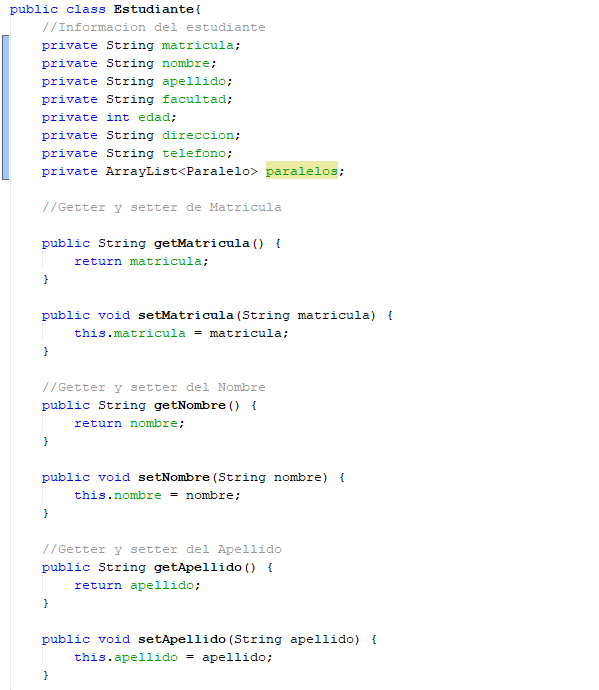
Técnica de refactorización.

Se coloca los atributos privados dentro de las clases y se utiliza el método **Change Bidirectional Association to Unidirectional** puesto que la clase Paralelo y Estudiante están relacionados mutuamente, pero en la clase Paralelo no se utiliza.

Código inicial.





Código refactorizado.