**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**SEGUNDO TÉRMINO 2020**

**DISEÑO DE SOFTWARE**

**Taller:**

Refactoring

**Integrantes:**

Kevin Chévez

Margarita Mawyin

Rafael Merchán.

**Fecha:** 07/01/2021

**Contenido**

[Sección A 5](#_Toc60951618)

[Code Smells identificados: 5](#_Toc60951619)

[1 Code Smell – Dispensables - Duplicate Code 5](#_Toc60951620)

[1.1 Consecuencias de mantener el código: 5](#_Toc60951621)

[1.2 Solución 5](#_Toc60951622)

[1.3 Captura inicial del código (Without Refactoring) 5](#_Toc60951623)

[1.4 Captura final del código (Refactoring Aplicated) 5](#_Toc60951624)

[2 Code Smell – Data Class and Temporary Field 6](#_Toc60951625)

[2.1 Consecuencias de mantener el código: 6](#_Toc60951626)

[2.2 Solución 6](#_Toc60951627)

[2.3 Captura inicial del código (Without Refactoring) 6](#_Toc60951628)

[2.4 Captura final del código (Refactoring Aplicated) 6](#_Toc60951629)

[3 Code Smell – Data Class and Feature Envy / Middle Man 7](#_Toc60951630)

[3.1 Consecuencias de mantener el código: 7](#_Toc60951631)

[3.2 Solución 7](#_Toc60951632)

[3.3 Captura inicial del código (Without Refactoring) 7](#_Toc60951633)

[3.4 Captura final del código (Refactoring Aplicated) 7](#_Toc60951634)

[4 Code Smell – (Dispensables -Data Class) 8](#_Toc60951635)

[4.1 Consecuencias de mantener el código: 8](#_Toc60951636)

[4.2 Solución 8](#_Toc60951637)

[4.3 Captura inicial del código (Without Refactoring) 8](#_Toc60951638)

[4.4 Captura final del código (Refactoring Aplicated) 8](#_Toc60951639)

[5 Code Smell – (Dispensables -Duplicate Code and Dead Code) 10](#_Toc60951640)

[5.1 Consecuencias de mantener el código: 10](#_Toc60951641)

[5.2 Solución 10](#_Toc60951642)

[5.3 Captura inicial del código (Without Refactoring) 10](#_Toc60951643)

[5.4 Captura final del código (Refactoring Aplicated) 12](#_Toc60951644)

[6 Code Smell – Dispensables-Comments 15](#_Toc60951645)

[6.1 Consecuencias de mantener el código: 15](#_Toc60951646)

[6.2 Solución 15](#_Toc60951647)

[6.3 Captura inicial del código (Without Refactoring) 15](#_Toc60951648)

[6.4 Captura final del código (Refactoring Aplicated) 15](#_Toc60951649)

[7 Code Smell – Couplers - Inappropriate Intimacy 17](#_Toc60951650)

[7.1 Consecuencias de mantener el código: 17](#_Toc60951651)

[7.2 Solución 17](#_Toc60951652)

[7.3 Captura inicial del código (Without Refactoring) 17](#_Toc60951653)

[7.4 Captura final del código (Refactoring Aplicated) 18](#_Toc60951654)

[8 Code Smell – (Dispensables -Data Class and Dead Code) 19](#_Toc60951655)

[8.1 Consecuencias de mantener el código: 19](#_Toc60951656)

[8.2 Solución 19](#_Toc60951657)

[8.3 Captura inicial del código (Without Refactoring) 19](#_Toc60951658)

[8.4 Captura final del código (Refactoring Aplicated) 20](#_Toc60951659)

[Sección B 22](#_Toc60951660)

[Link del repositorio: 22](#_Toc60951661)

**Objetivos**

* Identificar malos olores de programación en el código fuente adjunto y las técnicas de refactorización correspondientes.
* Aplicar técnicas de refactorización que pueden aplicarse para eliminar los malos olores previamente identificados.

**Descripción**

En equipos de trabajo, conformados por hasta tres estudiantes, (i) identifique los malos olores de programación encontrados en el código fuente adjunto, (ii) Identifique las técnicas de refactorización adecuadas para eliminar los malos olores encontrados y (iii) Refactorice el código fuente para obtener un código más limpio y fácil de leer. Justifique su respuesta.

**Especificaciones**

Considere un **sistema académico** que permite manejar los estudiantes registrados en ciertos paralelos de distintas materias. Además, cada materia tiene un profesor y podría tener asignado un ayudante.

**Sección A**

Elabore un **reporte** en el que **identifique los code smells** encontrados en el código adjunto. Para cada code smell debe **indicar el nombre**, las **consecuencias** de mantener el mismo en dicho código y la(s) técnicas de refactorización utilizadas para eliminarlo. Para cada mal olor coloque una captura inicial y una del código refactorizado. Indique cualquier asunción que realice. **[60%]**

**Sección B**

Cree un repositorio en GitHub para subir el reporte y realizar las mejoras sobre el código inicial y aplique las técnicas indicadas en el reporte. **[40%]**

**Entregables (enlace a repositorio en Github)**

1. Un **documento** (**disponible en formatos docx o pdf**) que contenga el reporte y las imágenes correspondientes. Además, el documento debe tener una carátula en la que se liste a los integrantes del equipo, una tabla de contenido. Cada code smell debe estar en una página.
2. El repositorio debe contener en el directorio raíz el documento indicado anteriormente y la carpeta del proyecto refactorizado. Subir a Sidweb el enlace a dicho repositorio.
3. Cada estudiante debe entregar una evaluación a los miembros de su equipo de trabajo usando la rúbrica de co-evaluación en la **tarea individual** “Co-evaluación Taller de Refactoring” a disposición en SidWeb.

**Rúbrica de Calificación**

|  |  |
| --- | --- |
| **Sección A** | **Valor** |
| Identificar al menos 6 malos olores | 30 |
| Identificar técnicas para eliminar dichos malos olores. | 30 |
| **Sección B** |  |
| Aplicar correctamente las técnicas de refactorización indicadas en el reporte | 40 |
| **Penalidades** | **Valor** |
| Penalidad por hora o fracción de hora de retraso en la entrega | -30 |
| No subir a SidWeb los entregables requeridos de acuerdo con lo especificado | -50 |
| No subir la rúbrica de coevaluación en tarea individual (afectación individual) | -100 |

## Sección A

Elabore un **reporte** en el que **identifique los code smells** encontrados en el código adjunto. Para cada code smell debe **indicar el nombre**, las **consecuencias** de mantener el mismo en dicho código y la(s) técnicas de refactorización utilizadas para eliminarlo. Para cada mal olor coloque una captura inicial y una del código refactorizado. Indique cualquier asunción que realice.

### Code Smells identificados:

#### Code Smell – Dispensables - Duplicate Code

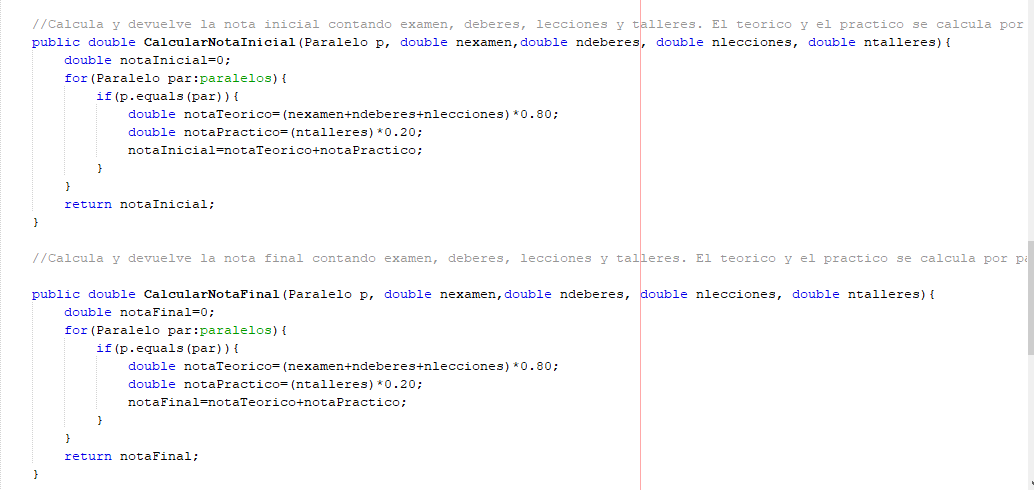
##### Consecuencias de mantener el código:

Dos métodos iguales, donde solo cambia el nombre del método, hacen más pesada la clase y más larga.

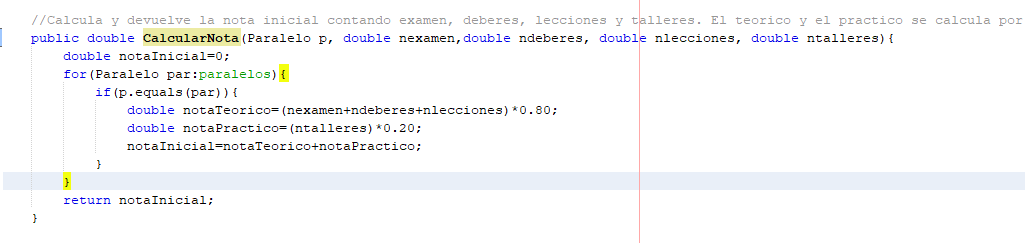
##### Solución

* Simplifying Method Calls - Parameterize Method
  + Dejar un solo método y cambiarle el nombre por uno más general. Al fin y al cabo, los dos métodos anteriores haciendo lo mismo, pero con diferentes. nombres.

##### Captura inicial del código (Without Refactoring)



##### Captura final del código (Refactoring Aplicated)



#### Code Smell – Data Class and Temporary Field

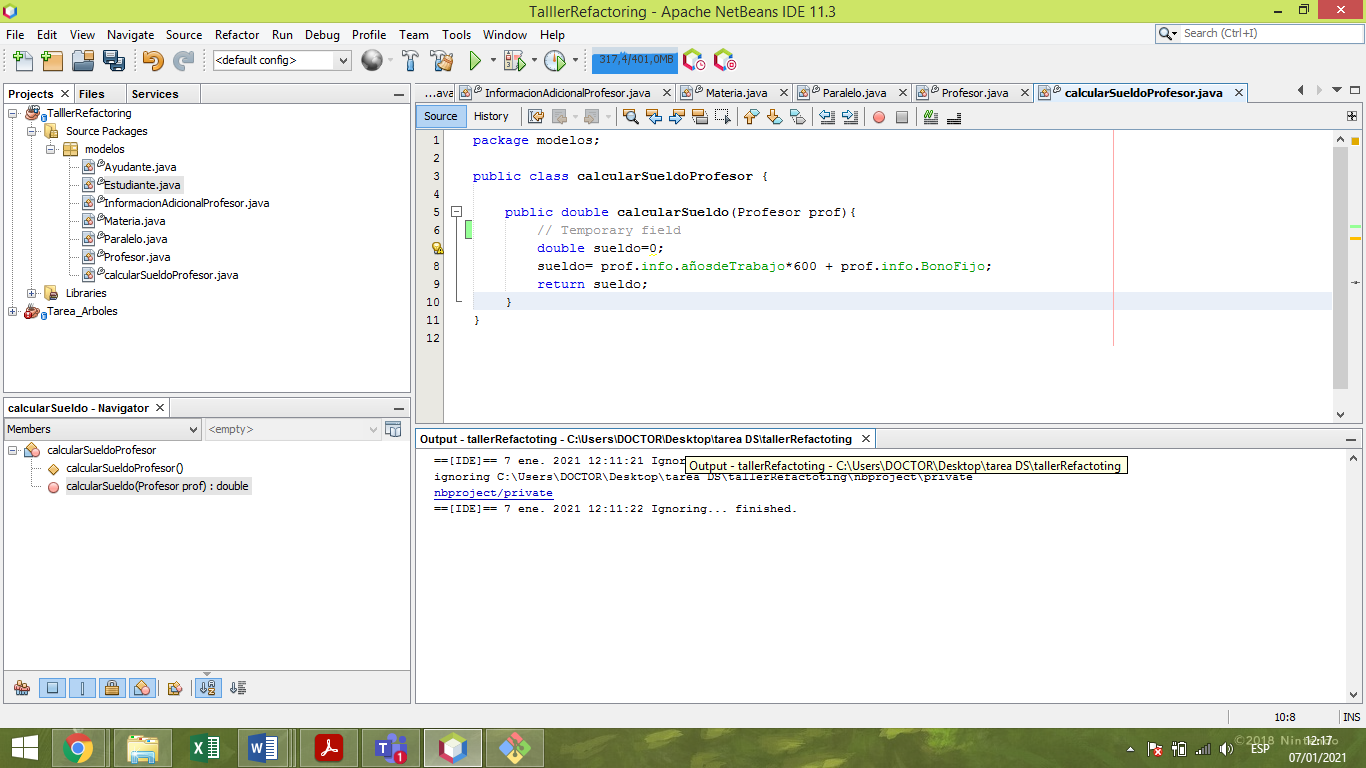
##### Consecuencias de mantener el código:

En la clase “calcularSueldoProfesor” está el método “calcularSueldo(Profesor prof)”. Este método, aparte de usar el Temporary Field “double sueldo = 0”, se lo puede colocar fácilmente en la clase Profesor.

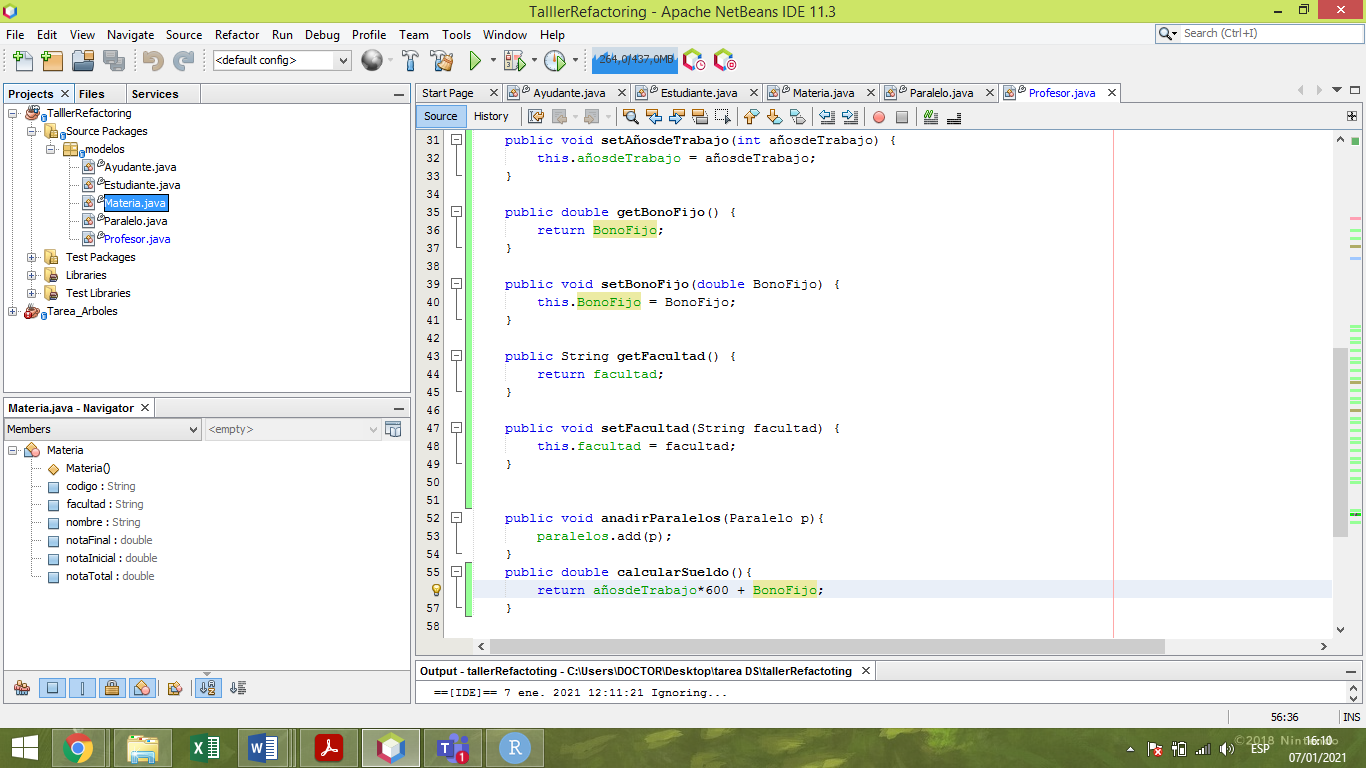
##### Solución

* Move Method
  + Se mueve el método para que esté dentro de la clase “Profesor” y se elimina la clase “calcularSueldoProfesor”

##### Captura inicial del código (Without Refactoring)



##### Captura final del código (Refactoring Aplicated)



#### Code Smell – Data Class and Feature Envy / Middle Man

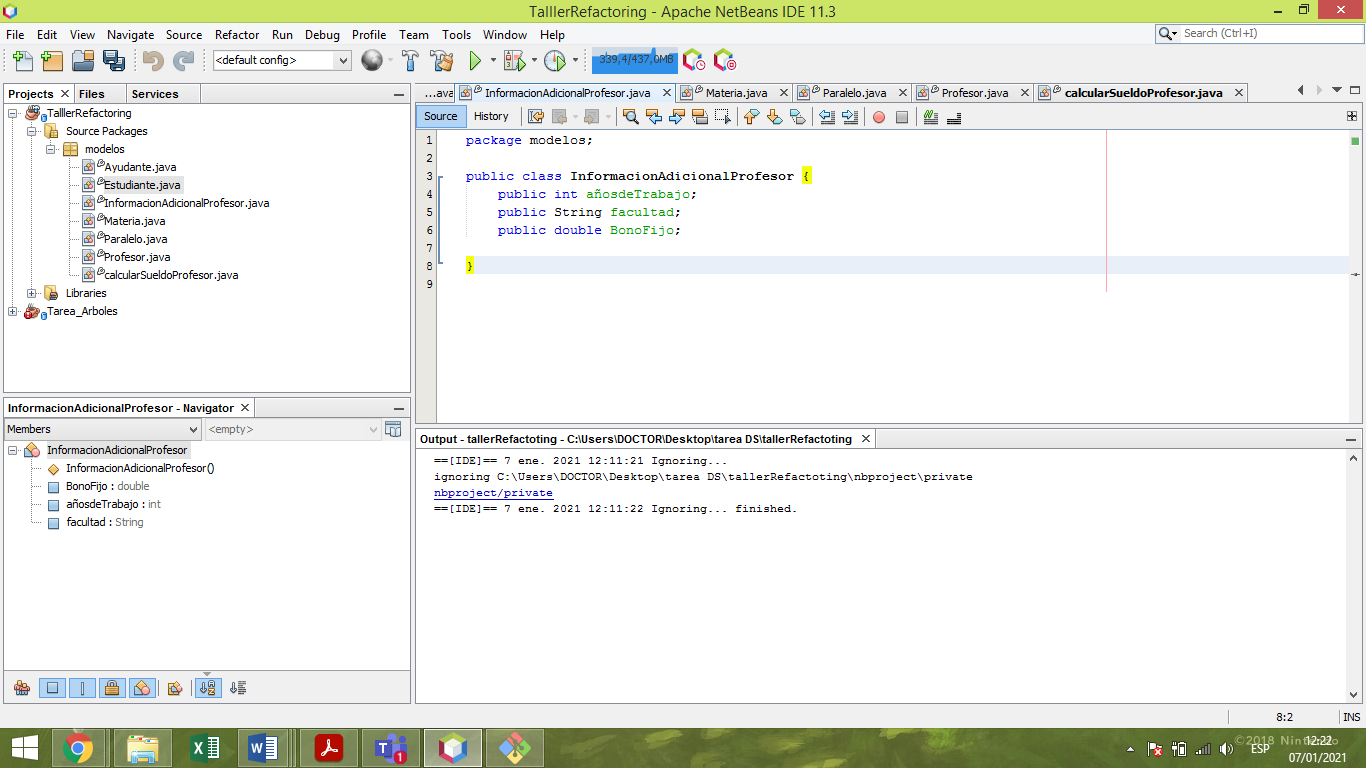
##### Consecuencias de mantener el código:

La clase de “InformacionAdicionalProfesor” es víctima de ser Feature Envy, debido a que no tiene ningún Getter ni Setter, de forma que solo se le agrega información con “profesor.info.atributodeInformacionAdicionalProfesor”. Esto puede ocasionar errores en el programa a largo plazo, en especial cuando se quiere acceder a algún dato en específico. Lo mejor que se puede hacer es poner todo en la clase Profesor.

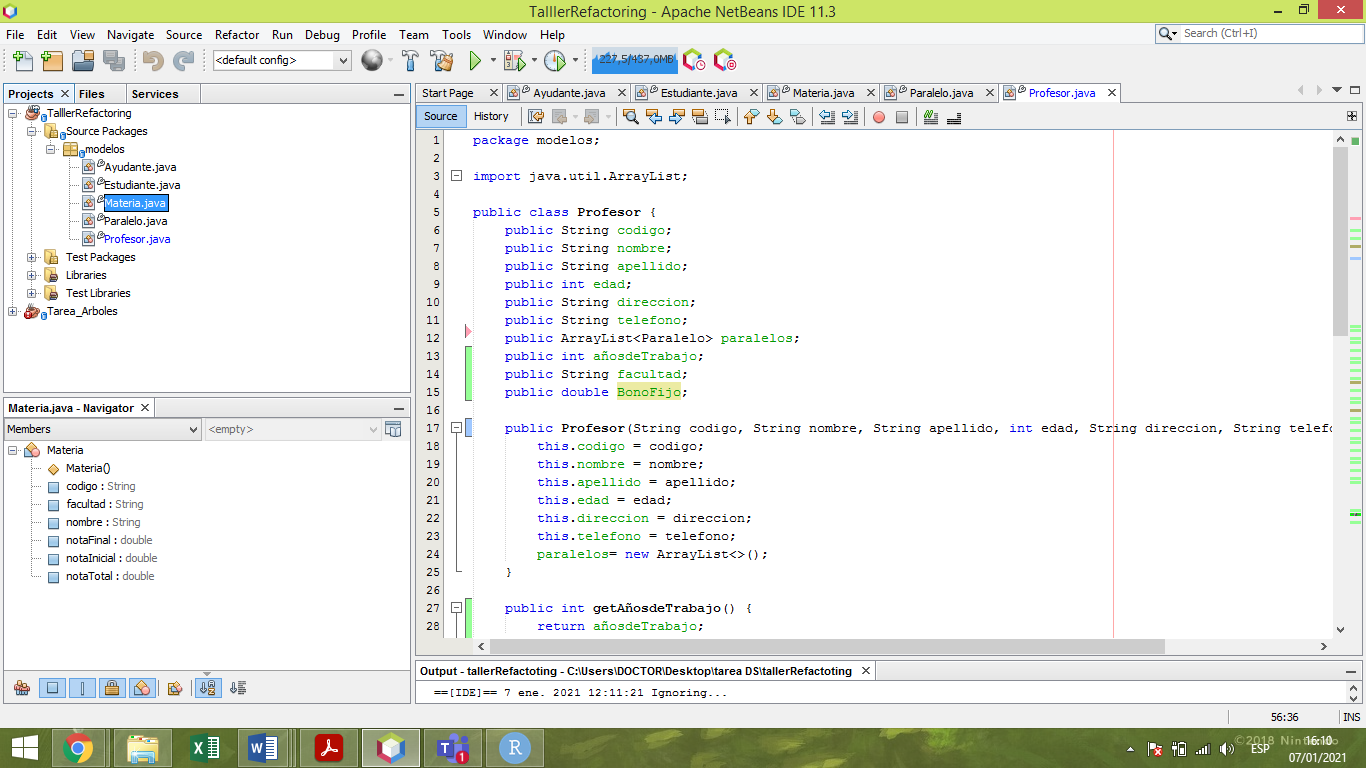
##### Solución

* Remove Middle Man
  + Se elimina la clase “InformacionAdicionalProfesor” y los atributos se los agrega a la clase “Profesor”. Además, se agregan Setters y Getters para esos atributos.

##### Captura inicial del código (Without Refactoring)



##### Captura final del código (Refactoring Aplicated)



#### Code Smell – (Dispensables -Data Class)

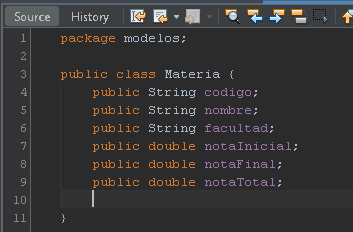
##### Consecuencias de mantener el código:

La clase Materia es una clase que aparenta solo ser usada como si fuera una clase de datos debido a que incumple con encapsulamiento, mantener esta clase de esta manera genera un costo innecesario al momento de mantener el sistema.

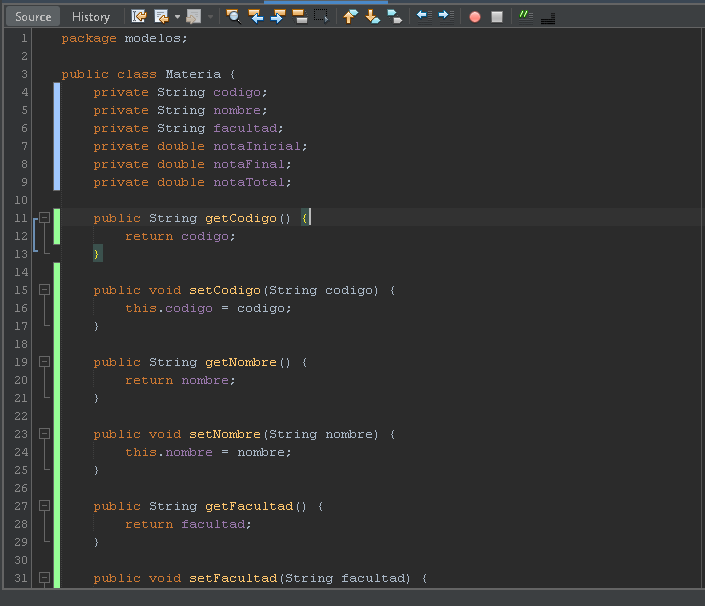
##### Solución

* Encapsulate Field
  + En este caso debemos aplicar principios de POO encapsulando los campos de la clase Materia

##### Captura inicial del código (Without Refactoring)



##### Captura final del código (Refactoring Aplicated)





#### Code Smell – (Dispensables -Duplicate Code and Dead Code)

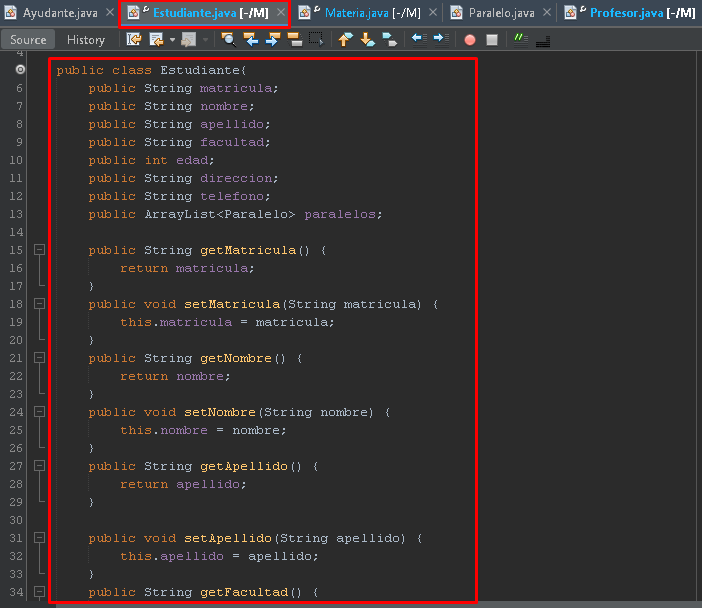
##### Consecuencias de mantener el código:

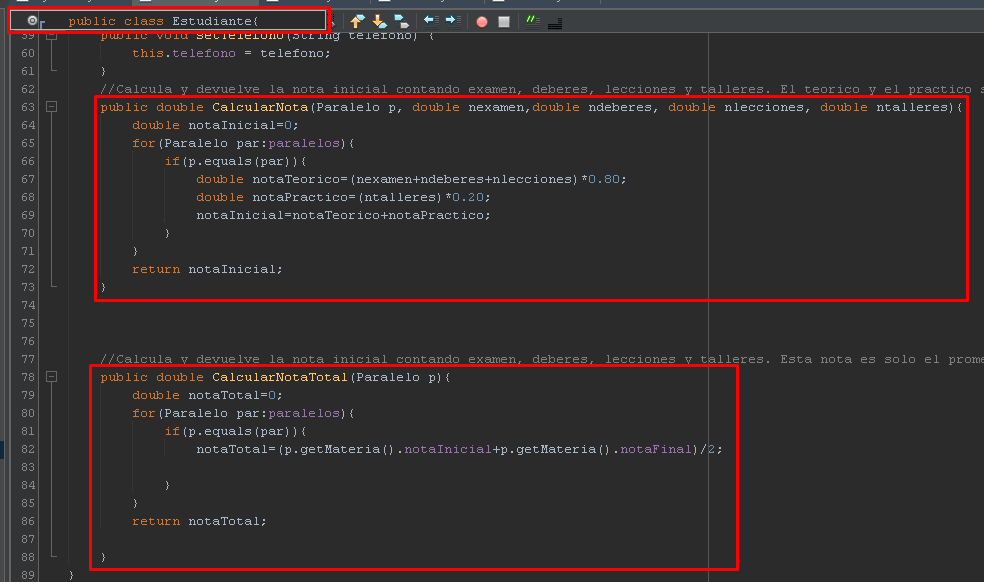
Mantener atributos/métodos duplicados generan costos innecesarios al momento de mantener un sistema, por ese motivo simplificar estas instancias de métodos o implementaciones de atributos mejoraría el mantenimiento del programa.

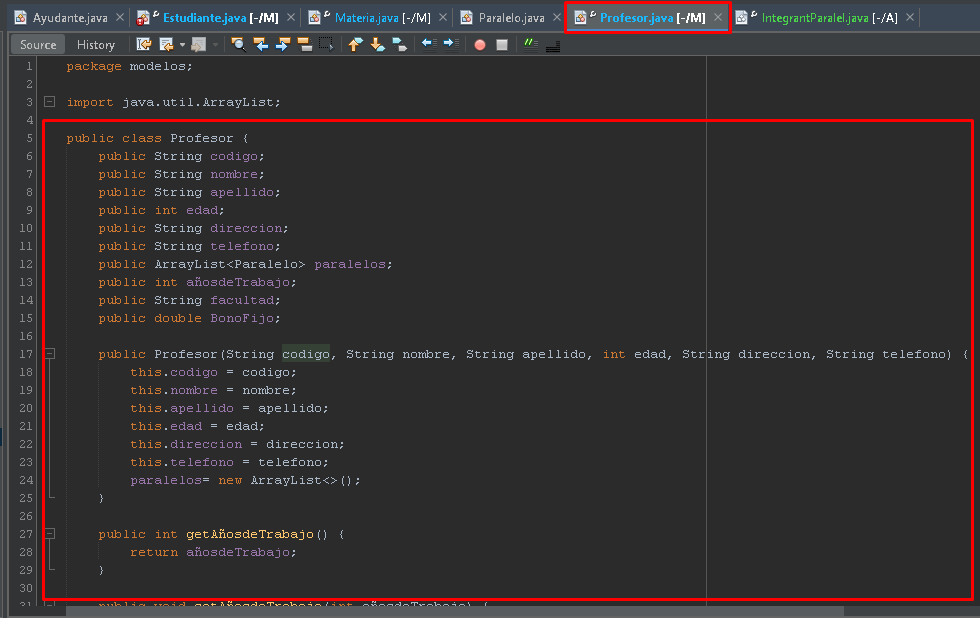
##### Solución

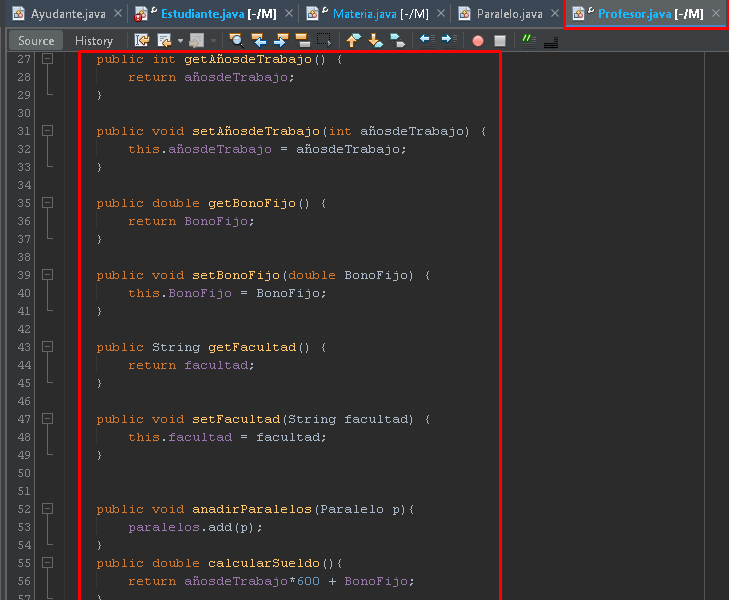
* Extract Superclass
  + Las clases Estudiante y Profesor son clases que contienen muchos atributos en común, estas pueden extender de una Superclase en común llamada IntegrantParalel.
* Mover características entre objetos – Subir Campo/Método
  + Los métodos que se encuentran en la clase Estudiante, CalcularNota y CalcularNotaTotal no deben ser manejados por un Estudiante sino por todas las clases que heredan de

##### Captura inicial del código (Without Refactoring)

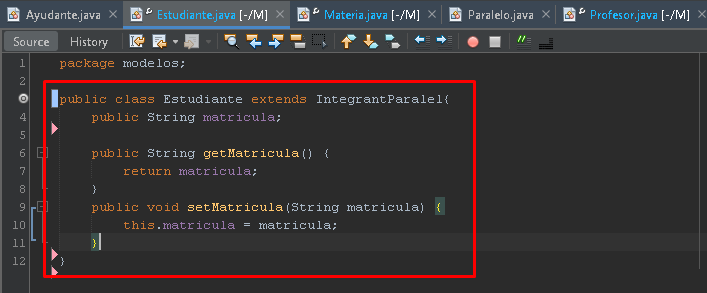


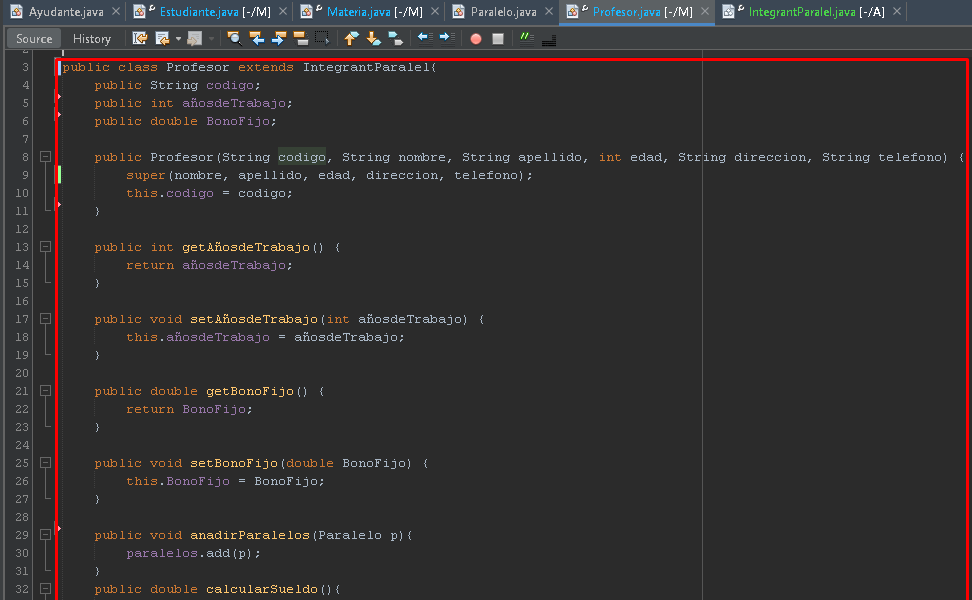


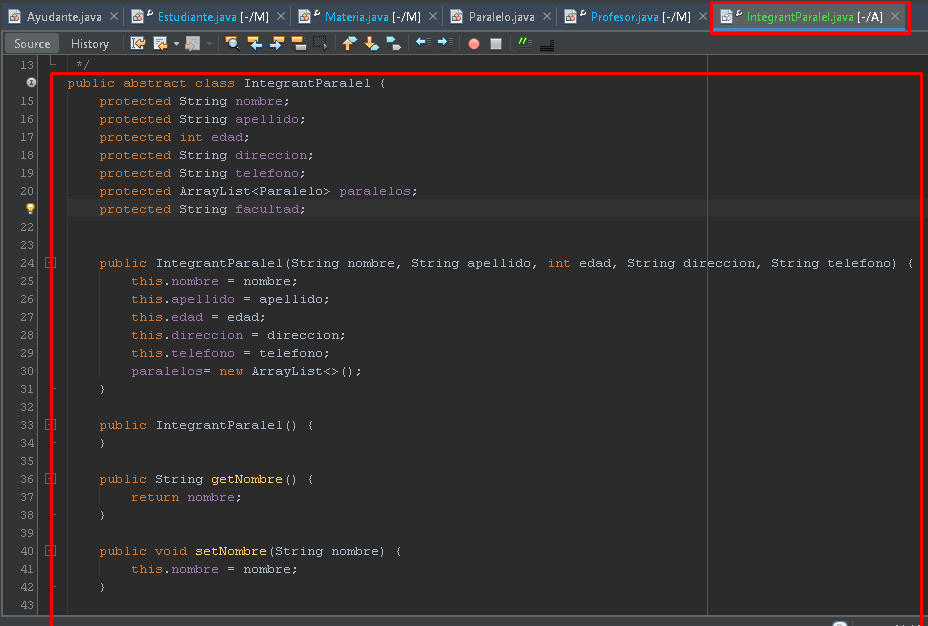


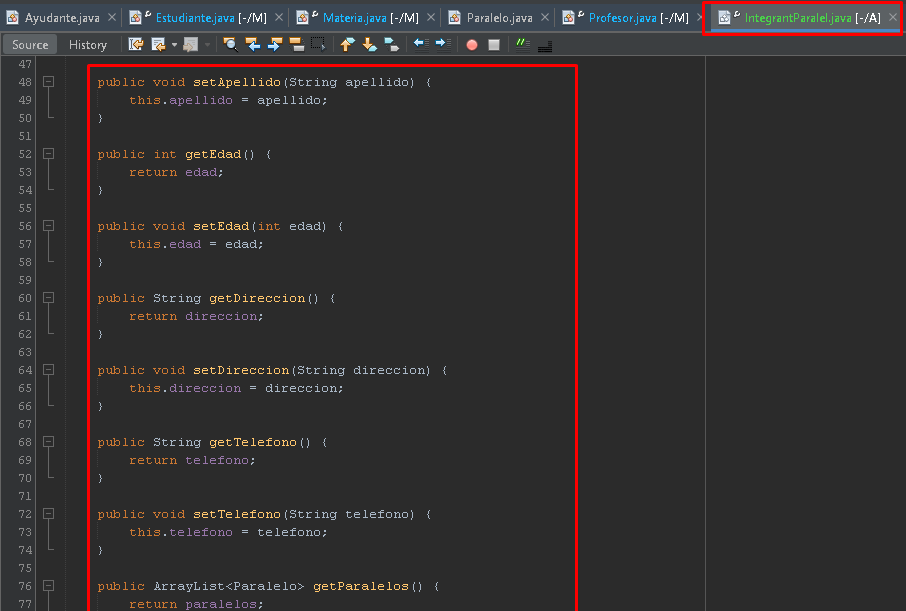


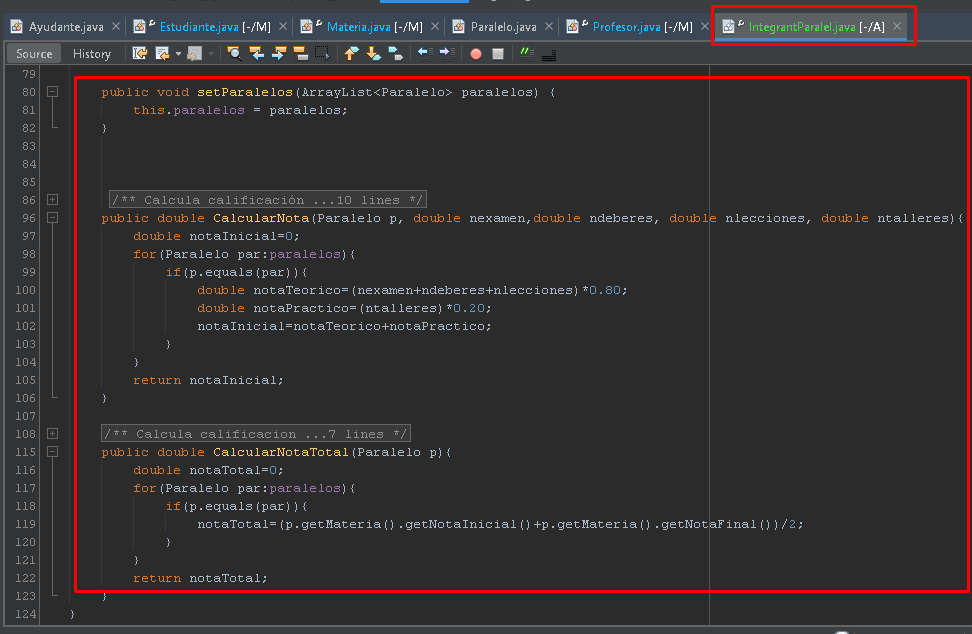
##### Captura final del código (Refactoring Aplicated)











#### Code Smell – Dispensables-Comments

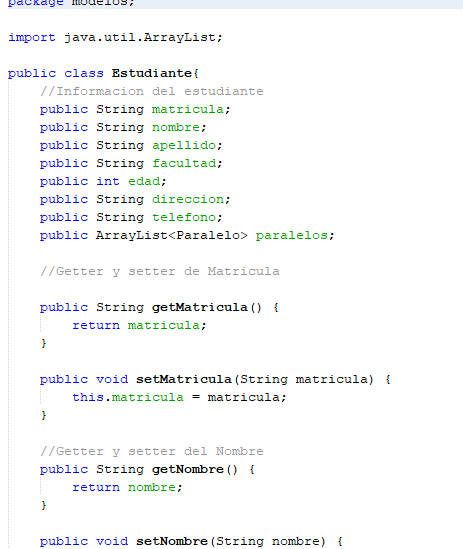
##### Consecuencias de mantener el código:

Eliminar los comentarios porque son obvios, aparte no hay comentarios para explicar algo que no se entienda.

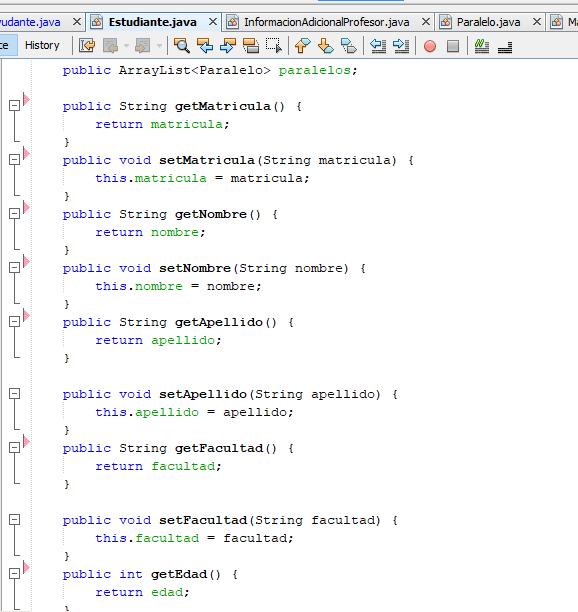
##### Solución

* Simplifying Method Calls - Rename Method
  + Borrar los comentarios obvios, el nombre del método habla más por su nombre, y si no, cambiar el nombre del método que haga entender que función realiza ese método.

##### Captura inicial del código (Without Refactoring)



##### Captura final del código (Refactoring Aplicated)



#### Code Smell – Couplers - Inappropriate Intimacy

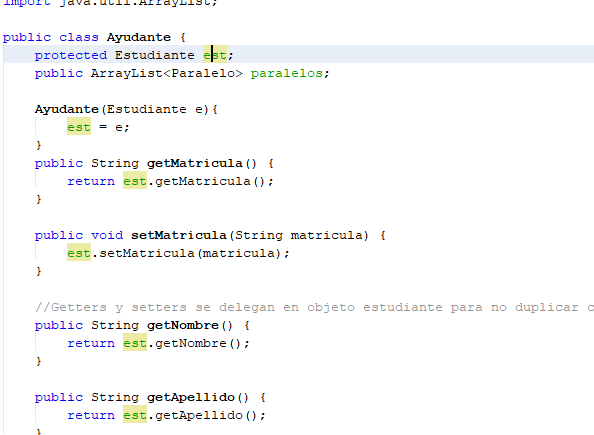
##### Consecuencias de mantener el código:

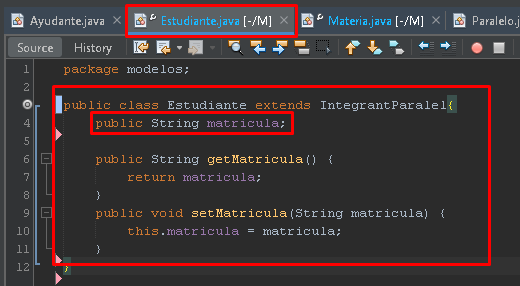
La clase Ayudante usa muchos atributos y métodos de la clase Estudiante, violando la privacidad de esa clase.

##### Solución

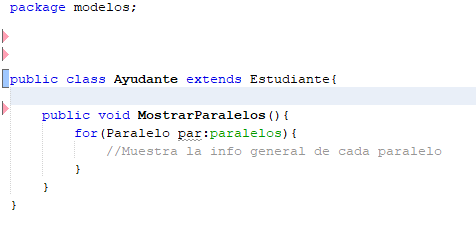
* Replace Delegation with Inheritance.
  + Hacer que la clase Ayudante extienda de la clase Estudiante, de esa manera usaria todos sus atributos sin violar la intimidad de otra clase, aparte puede tener sus propios metodos.
* Encapsulate Fields
  + Se tiene que revisar la superclase para aplicar principios de POO es decir, aplicar encapsulamiento a los atributos de la super clase.

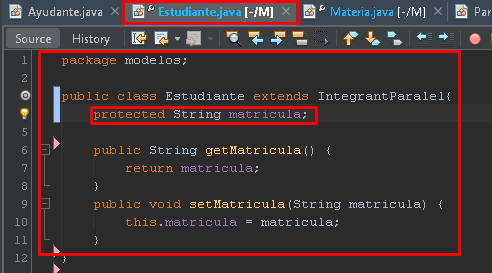
##### Captura inicial del código (Without Refactoring)





##### Captura final del código (Refactoring Aplicated)





#### Code Smell – (Dispensables -Data Class and Dead Code)

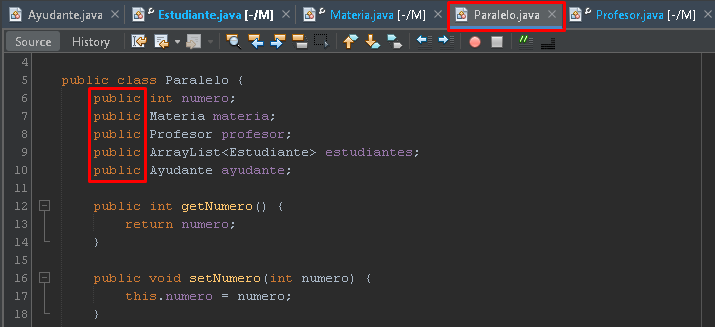
##### Consecuencias de mantener el código:

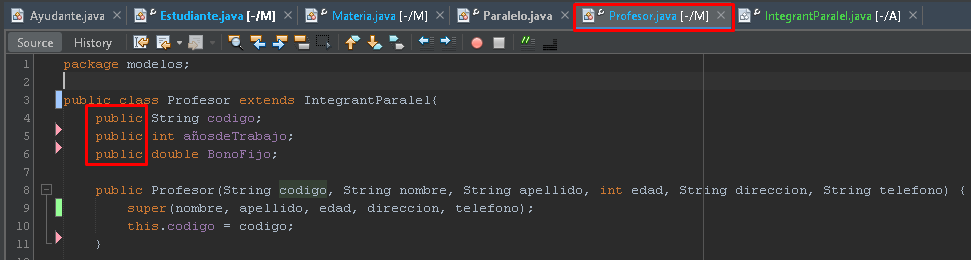
Mantener estas líneas de código inútiles genera más peso en la aplicación e ineficiencia, además que es evidente que una Data Clase viola la privacidad que debe tener toda clase y esto generaría problemas con la veracidad de los datos recopilados en la aplicación debido a que cualquier entidad podría controlar datos que deben ser privados.

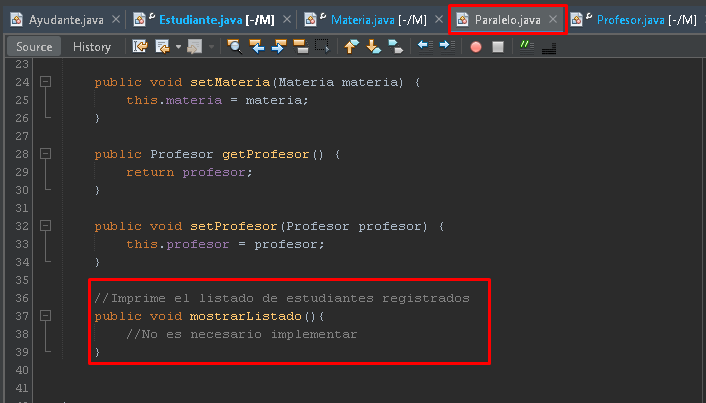
##### Solución

* Encapsulate Fields
  + Aplicar los principios de POO ayudan a mantener el programa eficiente y veraz con su información.
* Remove Parameters/Methods
  + Eliminar las líneas de códigos que son innecesarias disminuyen el costo que tiene un sistema en su mantenimiento.

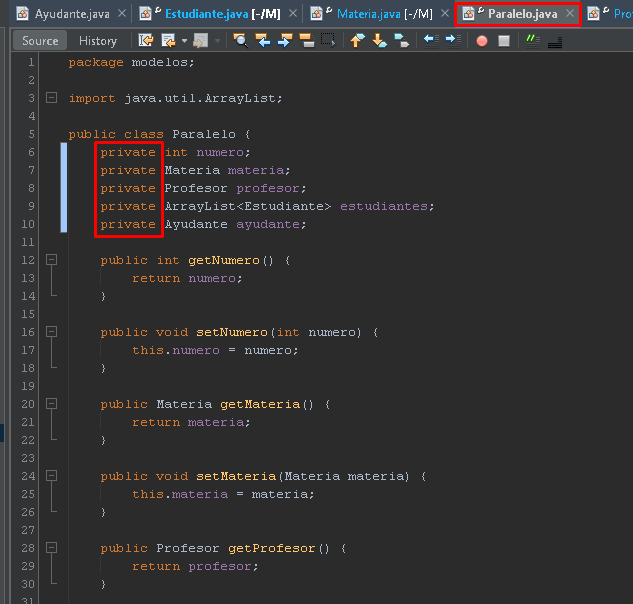
##### Captura inicial del código (Without Refactoring)

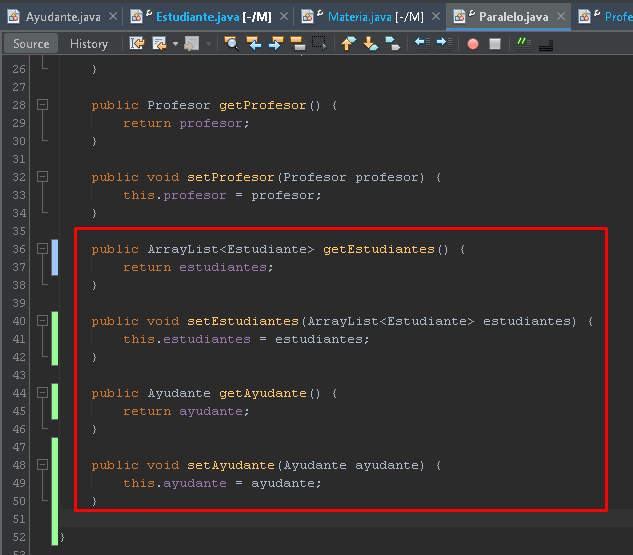


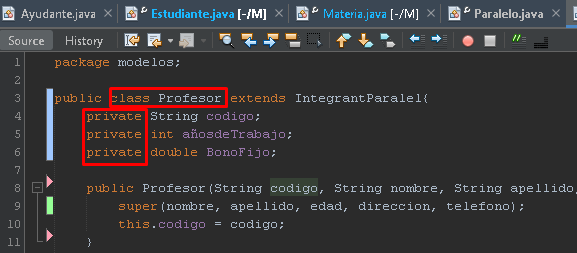


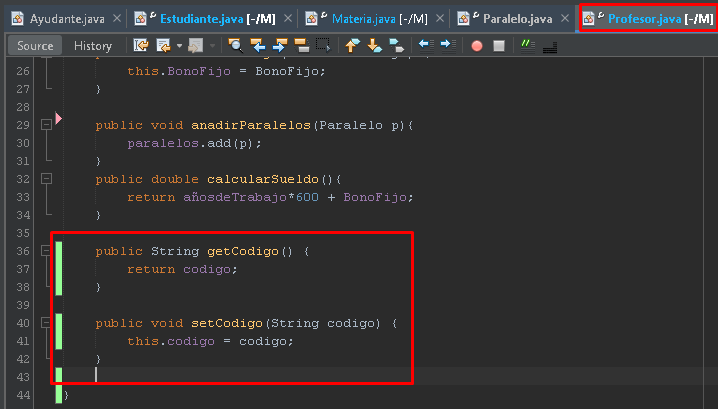


##### Captura final del código (Refactoring Aplicated)

(







## Sección B

Cree un repositorio en GitHub para subir el reporte y realizar las mejoras sobre el código inicial y aplique las técnicas indicadas en el reporte.

### Link del repositorio:

<https://github.com/MargaritaMawyin/tallerRefactoting.git>