

**DISEÑO DE SOFTWARE**

**Taller de REFACTORING - I TÉRMINO 2020**

**Profesor:** MSc. David Jurado

**Integrantes**  
María Camila Navarro Rey  
Joffre Emilio Ramírez Risco

Douglas Javier Sabando Macias

13 de Agosto del 2020

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**DISEÑO DE SOFTWARE**

**Taller de Refactoring - I TÉRMINO 2020**

# **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 |

**Contenido**

[Objetivos 2](#_Toc48246726)

[Descripción 2](#_Toc48246727)

[Especificaciones 2](#_Toc48246728)

[Entregables (enlace a repositorio en Github) 2](#_Toc48246729)

[Rúbrica de Calificación 2](#_Toc48246730)

[Temporary Field 5](#_Toc48246731)

[CONSECUENCIAS 5](#_Toc48246732)

[TÉCNICAS DE REFACTORIZACIÓN 5](#_Toc48246733)

[CÓDIGO INICIAL 5](#_Toc48246734)

[CÓDIGO MEJORADO 5](#_Toc48246735)

[Lazy Class 6](#_Toc48246736)

[CONSECUENCIAS 6](#_Toc48246737)

[TECNICAS DE REFACTORIZACION 6](#_Toc48246738)

[CÓDIGO INICIAL 6](#_Toc48246739)

[CÓDIGO MEJORADO 7](#_Toc48246740)

[Feature Envy 8](#_Toc48246741)

[CONSECUENCIAS 8](#_Toc48246742)

[TECNICAS DE REFACTORIZACION 8](#_Toc48246743)

[CÓDIGO INICIAL 8](#_Toc48246744)

[CÓDIGO MEJORADO 9](#_Toc48246745)

[Duplicate Code 10](#_Toc48246746)

[CONSECUENCIAS 10](#_Toc48246747)

[TECNICAS DE REFACTORIZACION 10](#_Toc48246748)

[CODIGO INICIAL 10](#_Toc48246749)

[CÓDIGO MEJORADO 11](#_Toc48246750)

[Long Method/Parameter 12](#_Toc48246751)

[CONSECUENCIAS 12](#_Toc48246752)

[TECNICAS DE REFACTORIZACION 12](#_Toc48246753)

[CÓDIGO INICIAL 12](#_Toc48246754)

[CÓDIGO MEJORADO 12](#_Toc48246755)

[Large Class 13](#_Toc48246756)

[CONSECUENCIAS 13](#_Toc48246757)

[TECNICAS DE REFACTORIZACION 13](#_Toc48246758)

[CÓDIGO INICIAL 13](#_Toc48246759)

[CÓDIGO MEJORADO 14](#_Toc48246760)

# **Temporary Field**

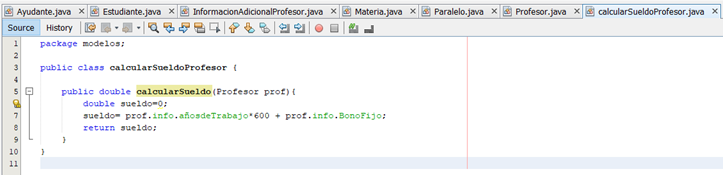
Se encontró este code smell dentro del metodo calcularSueldoProfesor, la variable sueldo está siendo usada temporalmente para guardar el dato del sueldo, mientras que se puede omitir ese proceso y reornar directamente la operación.

## **CONSECUENCIAS** La variable del método calcularSueldoProfesor ocupando recurso de nuestro sistema, el tiempo de ejecución de nuestro programa se verá ligeramente afectado. Complica la comprensión del código.

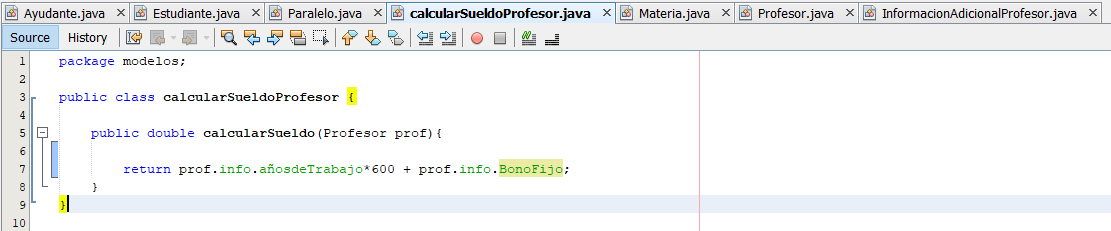
## **TÉCNICAS DE REFACTORIZACIÓN**

**Replace Method with Method Object:** La clase calcularSueldo usa una variable temporal llamada sueldo y se hace return de esta. Como solución eliminamos la variable temporal y se retorna la operación directamente.

## CÓDIGO INICIAL



## CÓDIGO MEJORADO



# **Lazy Class**

Se encontraron dos casos de Lazy class en InformacionAdicionalProfesor y en CambiarSueldoProfesor.

# **CONSECUENCIAS**

No es conveniente dejar una clase no está haciendo lo suficiente, ya que en si no está haciendo nada más que ocupar espacio y memoria por ende es necesario eliminarla o aumentar su responsabilidad.

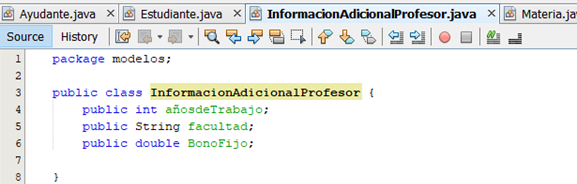
## **TECNICAS DE REFACTORIZACION**

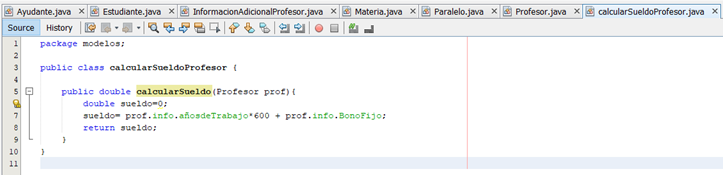
**Inline Class:** Mover todas las funciones y variables de la clase a otra.

Esto se lo puede hacer de dos maneras la primera es moviendo los datos de la clase InformacionAdicionalProfesor y el método calcularSueldo a la clase Profesor.

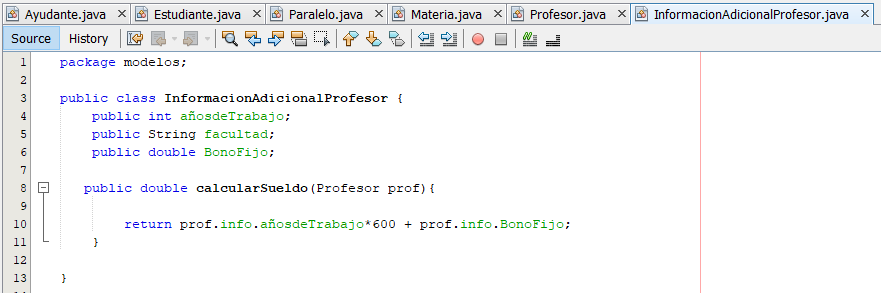
La segunda manera es mover el método calcularSueldo a InformaciónAdicionalProfesor y asi ambos dejan de ser clases vagas.

## CÓDIGO INICIAL





## CÓDIGO MEJORADO



# **Feature Envy**

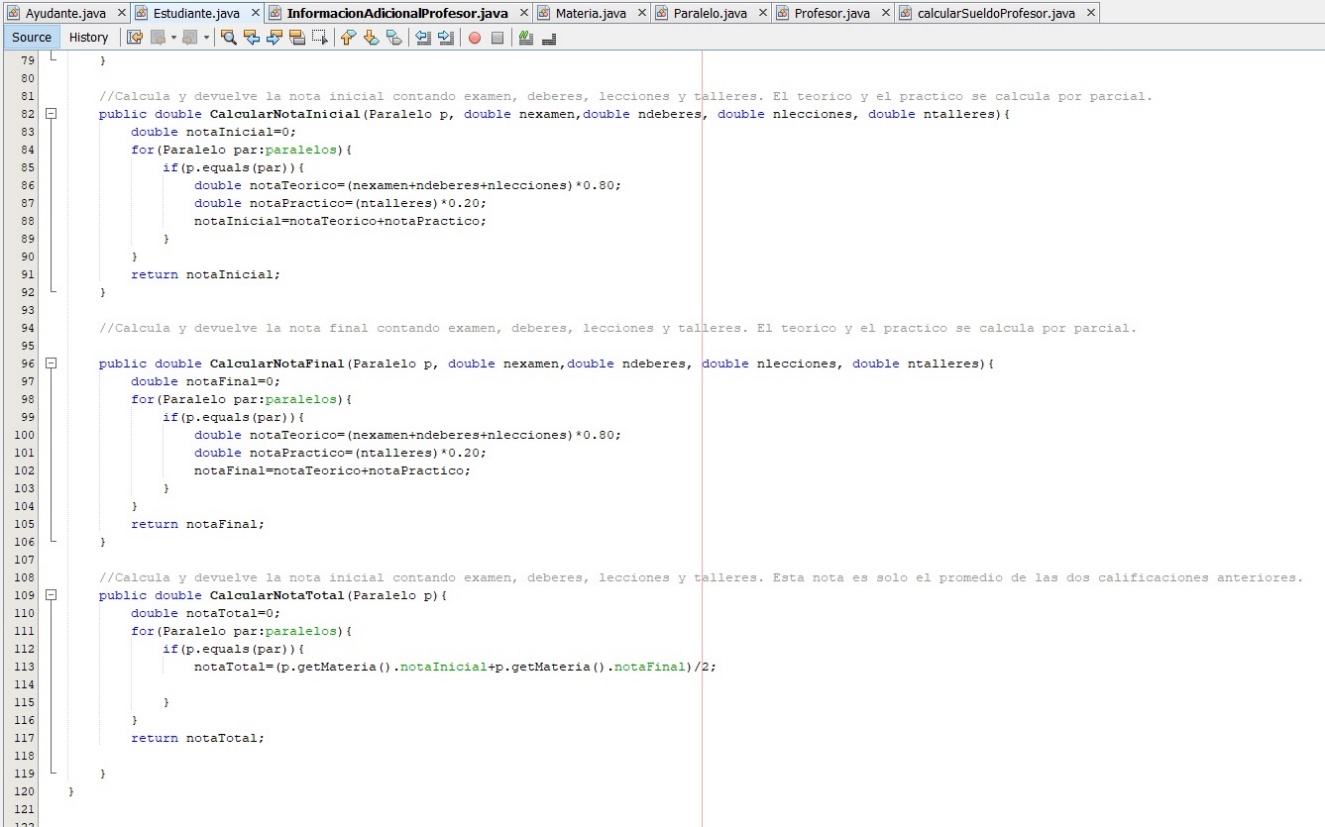
## **CONSECUENCIAS**

Duplicación de código, mala organización del código.

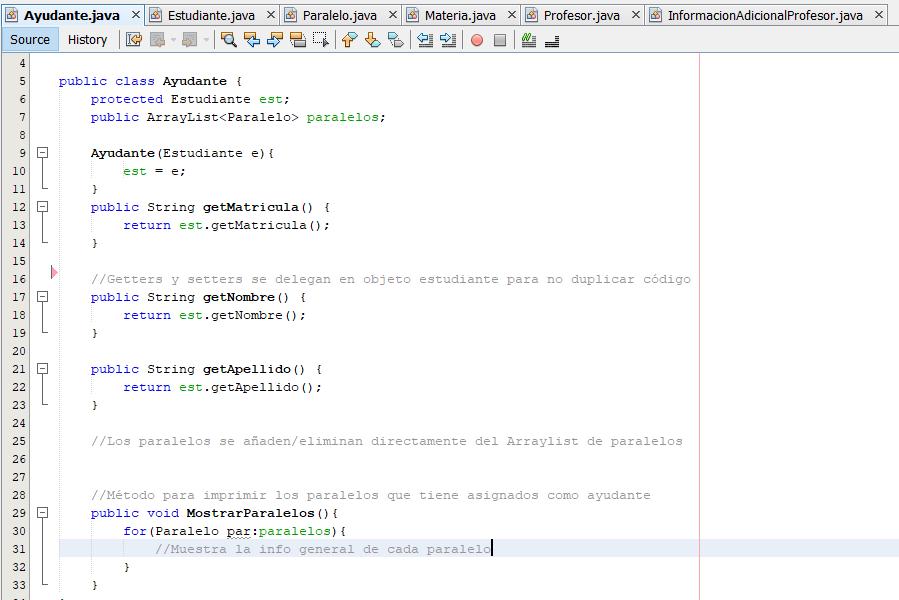
## **TECNICAS DE REFACTORIZACION**

La clase ayudante no debería tener el metodo setMatricula(), porque este no debería poder cambiar la matrícula de otro estudiante. Para solucionar esto usamos **RemoveSettingMethod**, que consiste en remover dicho método.

## CÓDIGO INICIAL



## CÓDIGO MEJORADO



# **Duplicate Code**

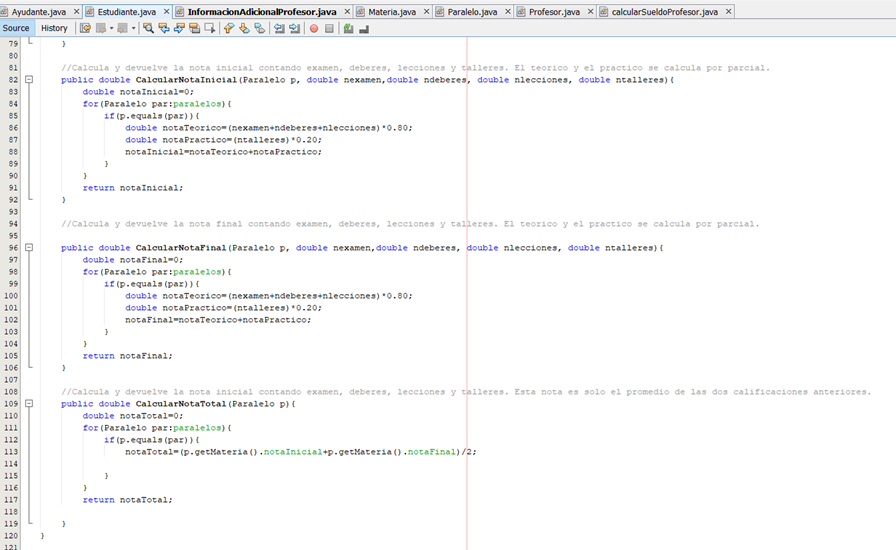
## **CONSECUENCIAS**

El código duplicado con frecuencia crea largas secciones repetidas de código que difieren en sólo unas pocas líneas o caracteres. La longitud de estas rutinas puede hacerlo difícil de comprenderlas rápidamente.

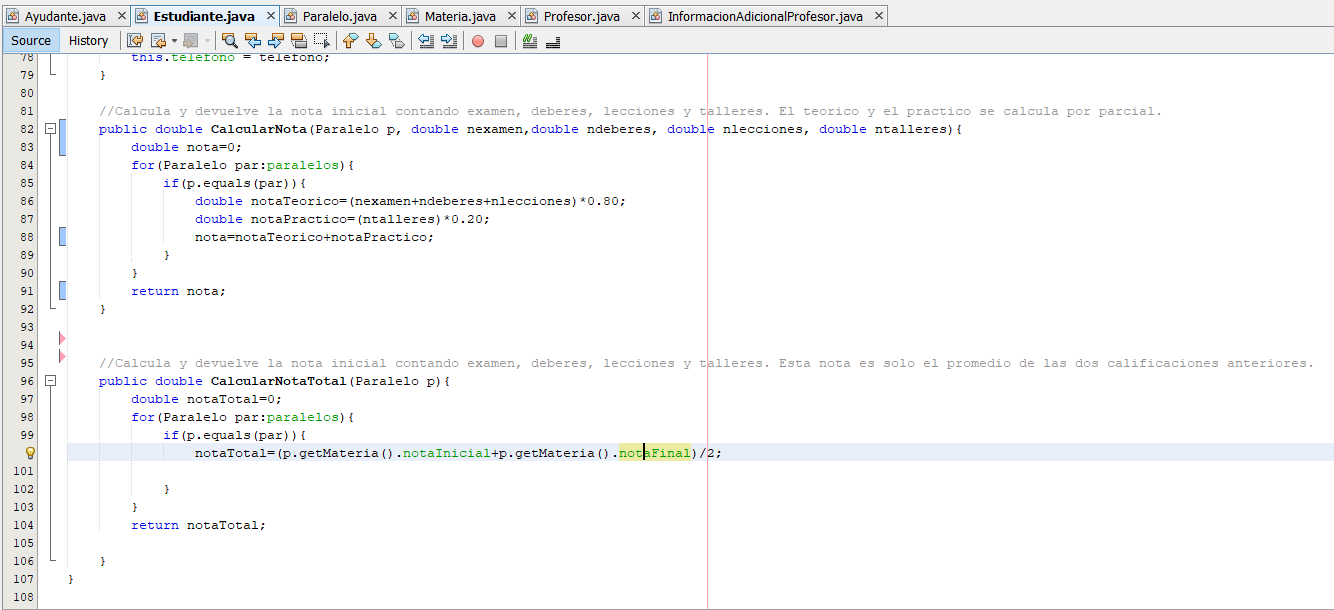
## **TECNICAS DE REFACTORIZACION**

Para corregir este smell code hemos usado la técnica de **Substitute Algorithm** en donde utilizamos el método más óptimo de los cuales posean el código dúplicado, en nuestro caso simplemente eliminaremos el método **calcularNotaInicial** ya que haccen exactamente lo mismo que **calcularNotaFinal** y este último lo dejaremos como un método general lllamado calcularNota.

## CODIGO INICIAL



## CÓDIGO MEJORADO



# **Long Method/Parameter**

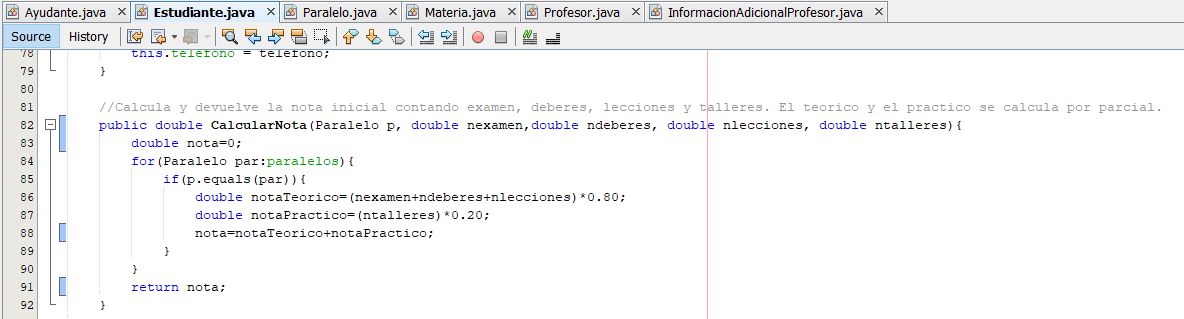
## **CONSECUENCIAS**

Puede generar errores al introducir los parametros al metodo correspondiente.

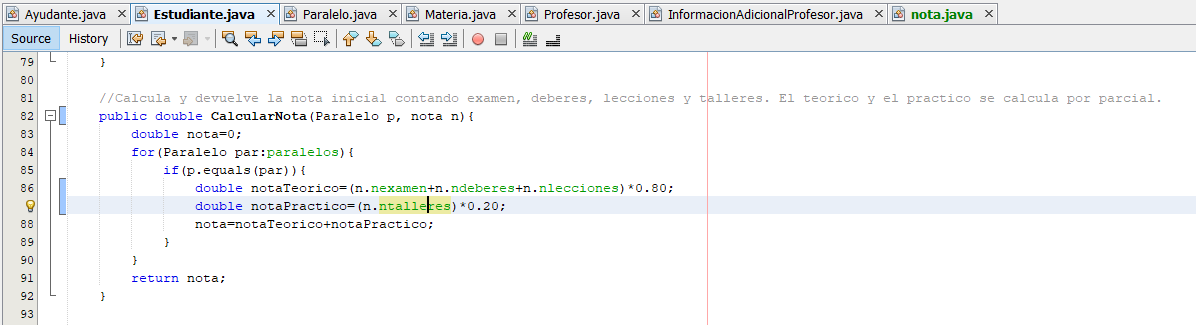
## **TECNICAS DE REFACTORIZACION**

**Preserve Whole Object** con el cual se redujo el número de argumentos creando un objeto llamado nota para eviar la saturación de parametros, de esta manera el codigo sera mas facil de leer y de trabajar.

## CÓDIGO INICIAL



## CÓDIGO MEJORADO



# **Large Class**

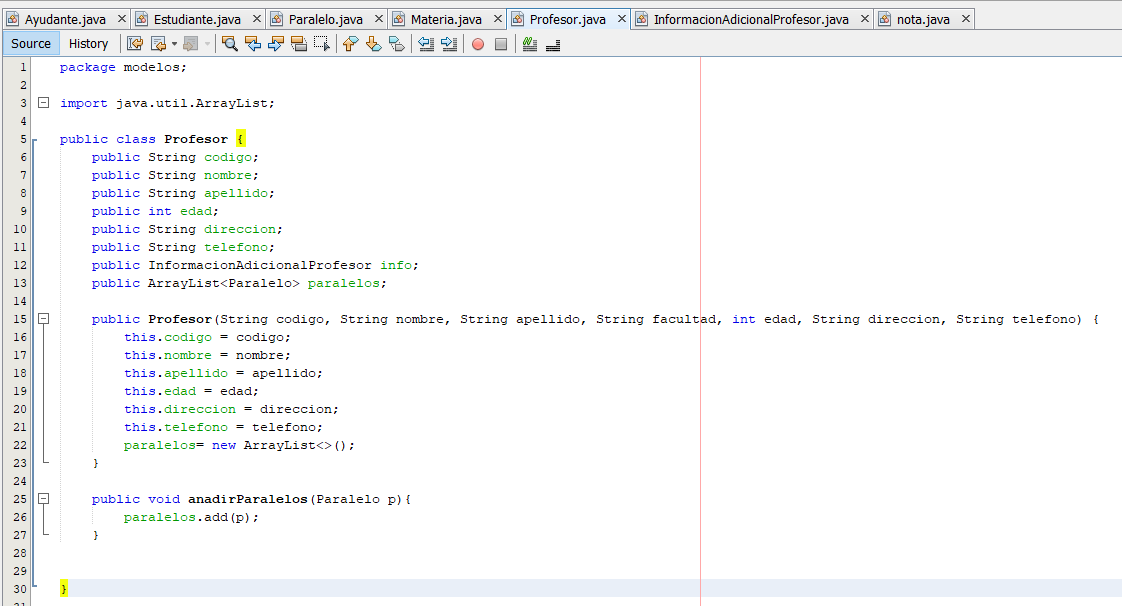
## **CONSECUENCIAS**

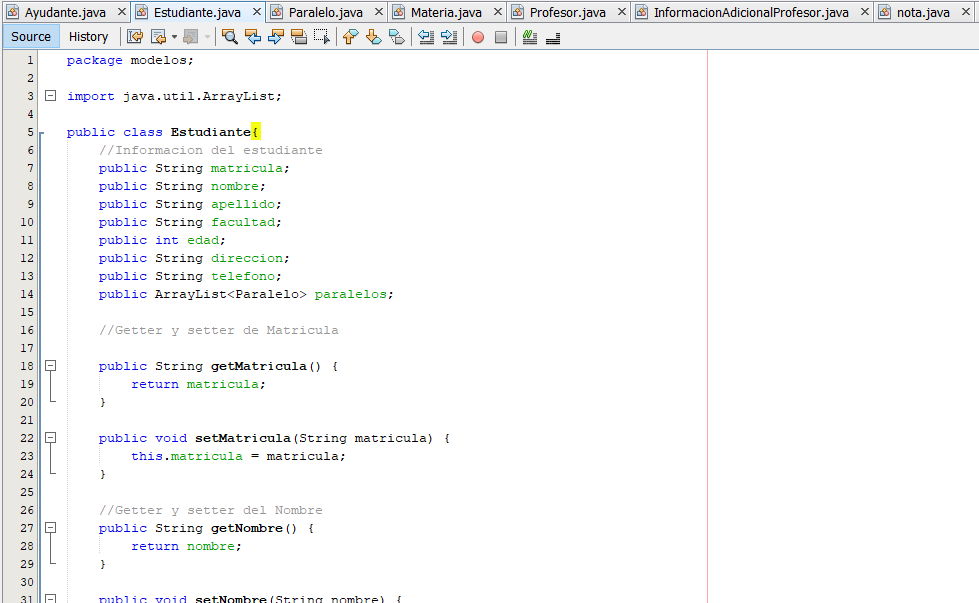
El código se vuelve difícil de entender y soportar, por la gran cantidad de atributos que posee.

## **TECNICAS DE REFACTORIZACION**

Usamos **Extract Class** para que así tanto como estudiante y profesor compartan atributos heredandolos de una clase padre que se llamará “Persona”.

## CÓDIGO INICIAL





## CÓDIGO MEJORADO

