

Materia: Diseño de Software

Taller Refactoring

Grupo # 9

Integrantes:

* Alex Villegas
* Ysrael Larco
* Guillermo Veintimilla

Contenido

[Code Smell: Data Class 3](#_Toc60954690)

[Code Smell: Inappropiate intimacy 4](#_Toc60954691)

[Code Smell: Data Clumps 5](#_Toc60954692)

[Code Smell: Temporary Field 6](#_Toc60954693)

[Code Smell: Primitive Obsession 7](#_Toc60954694)

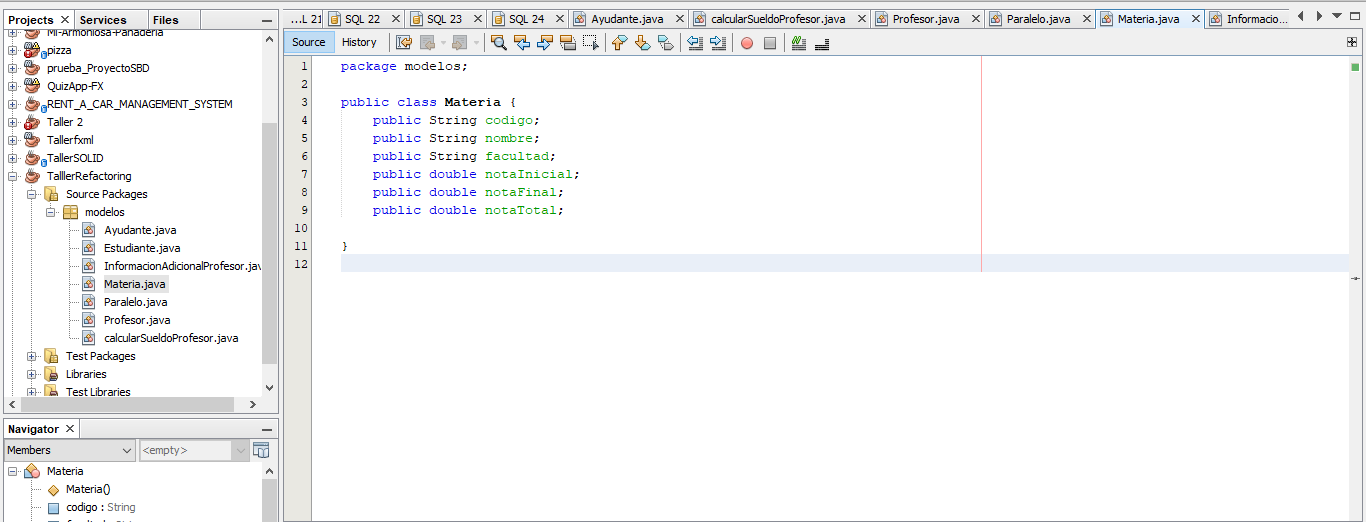
[Code Smell: Lazy class 8](#_Toc60954695)

# Code Smell: Data Class

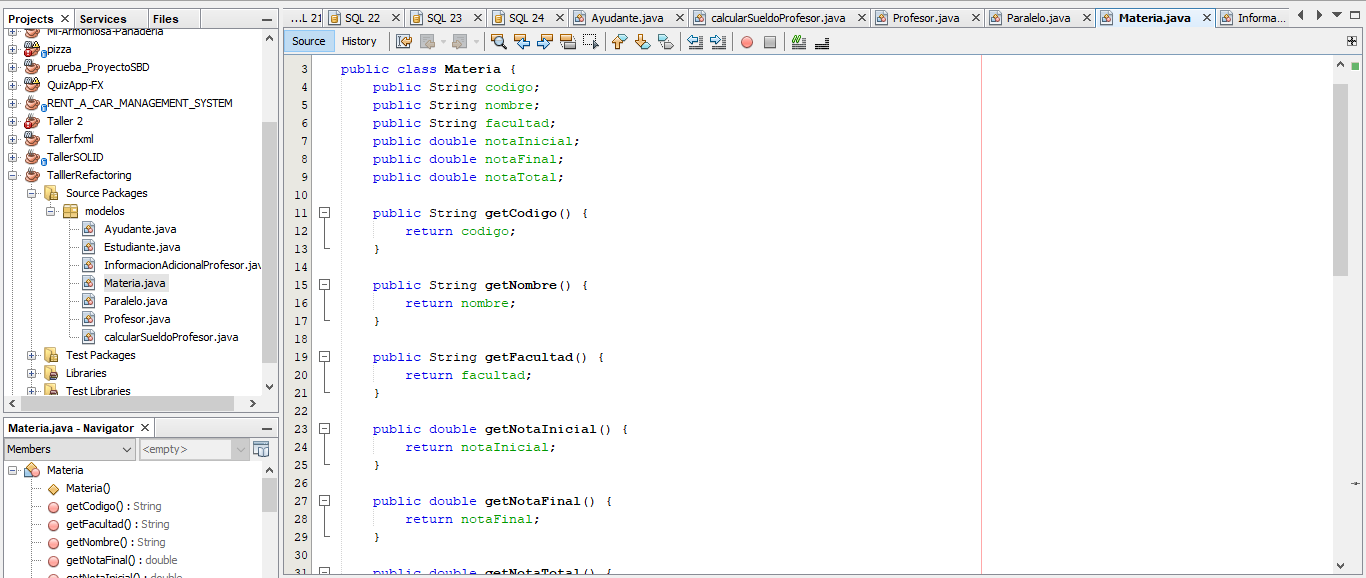
**Consecuencias:** En sí, mantener esta clase no afecta al funcionamiento normal del programa, sin embargo, se vuelve inútil al no contener ni getters ni setters o comportamiento alguno. Por lo tanto, estos deberían ser implementados ya que una característica principal de los objetos es el poder contener tipos de comportamientos u operaciones.

**Técnica de Refactorización para utilizar:** Encapsulate Field

Con esta técnica se estarían creando los getters y setters para la clase “Materia” con el fin de que exista encapsulación para que puedan ocultar sus datos de otros objetos y con esto, el código se vuelve más fácil de mantener.

Código anterior:

Código actualizado:



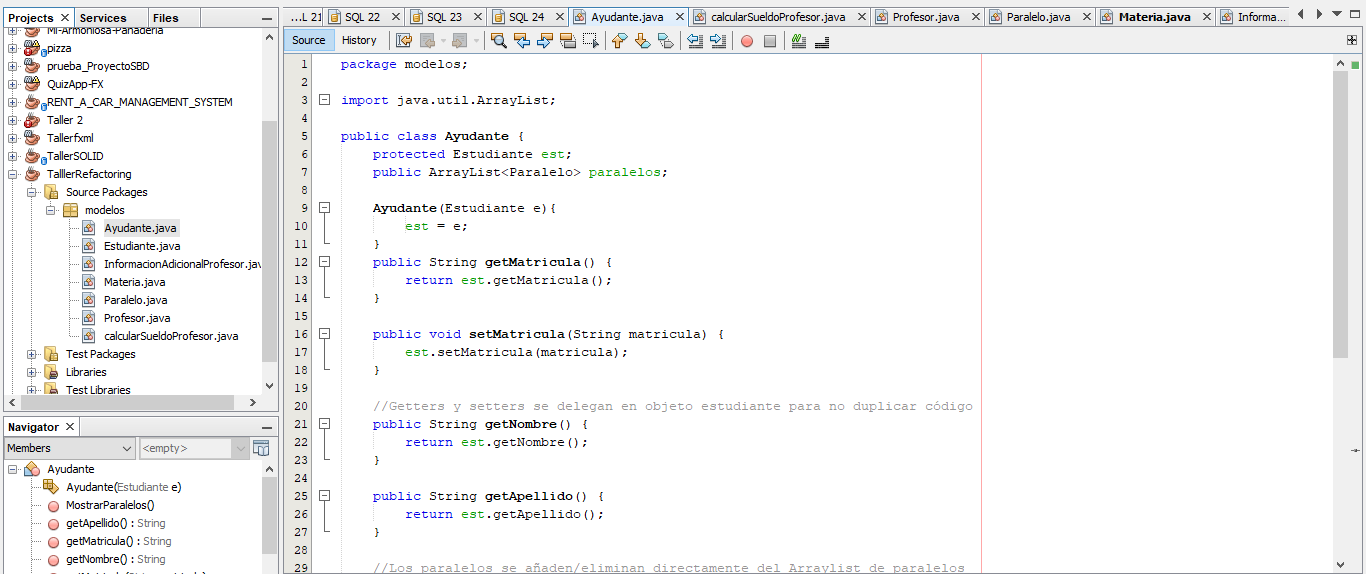
# Code Smell: Inappropiate intimacy

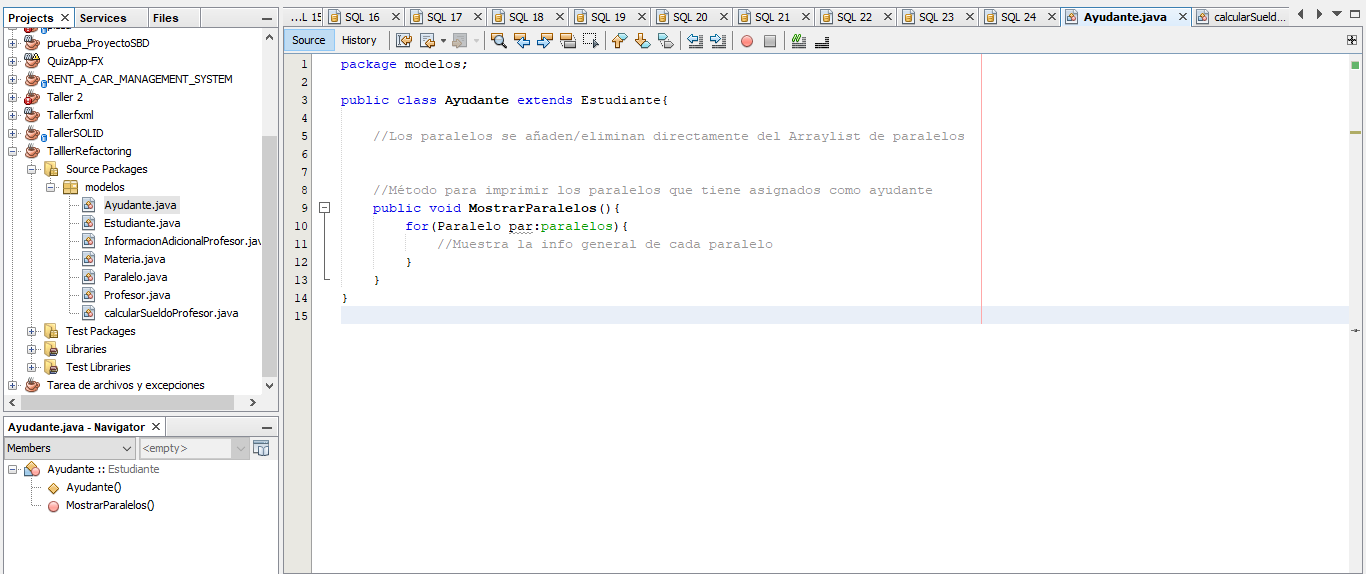
**Consecuencias:** En este caso, la clase “Ayudante” al tener como atributo un objeto de la clase “Estudiante”, accede a los datos sensibles de “Estudiante” y provoca que exista un gran acoplamiento entre las dos, es decir, si se realiza un cambio en “Estudiante” también se tendrán que aplicar cambios a “Ayudante”. Así, las clases se vuelven difíciles de reutilizar y mantener.

**Técnica de Refactorización para utilizar:** Replace delegation with inheritance

Esta técnica permitiría que la clase “Ayudante” herede de la clase “Estudiante” haciendo innecesarios los métodos de delegación en “Ayudante” y reduciendo el tamaño del código.

Código anterior:



Código actualizado:

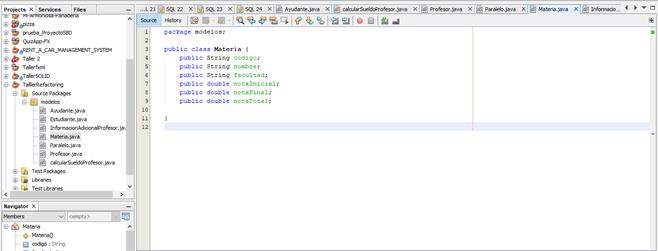
# Code Smell: Data Clumps

**Consecuencias:** Tener código repetido e idéntico en distintas partes del código representa problemas ya que se tiene una clase innecesariamente larga, y como no se asegura si algunas variables tienen relación entre si esto puede provocar inconvenientes al momento de manipularlas.

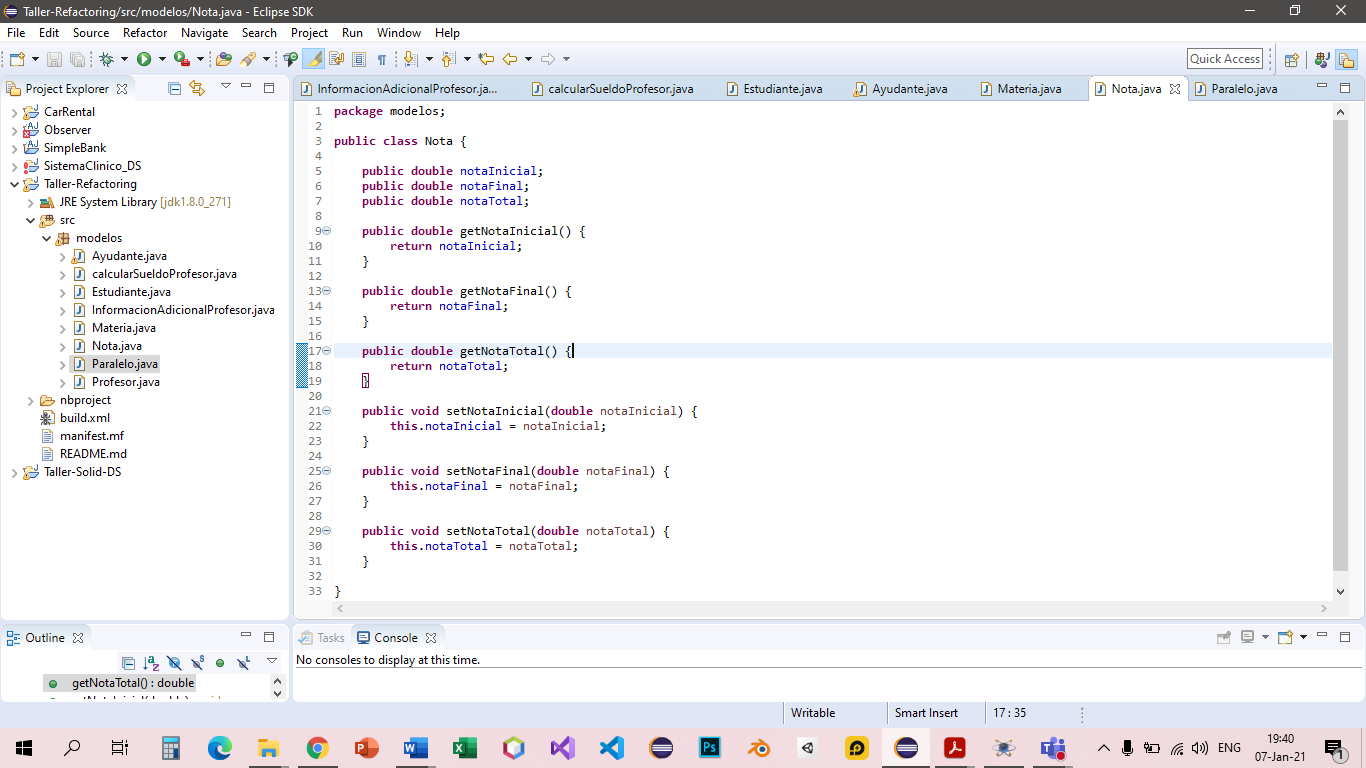
**Técnica de Refactorización para utilizar:** Extrac Class

Esta técnica permitiría separar la variable notainicial y notafinal en su propia clase asi se podrá editarla de una manera más eficiente sin que este presente problemas estando en otra clase como variable.

Codigo Anterior:



Codigo Nuevo:



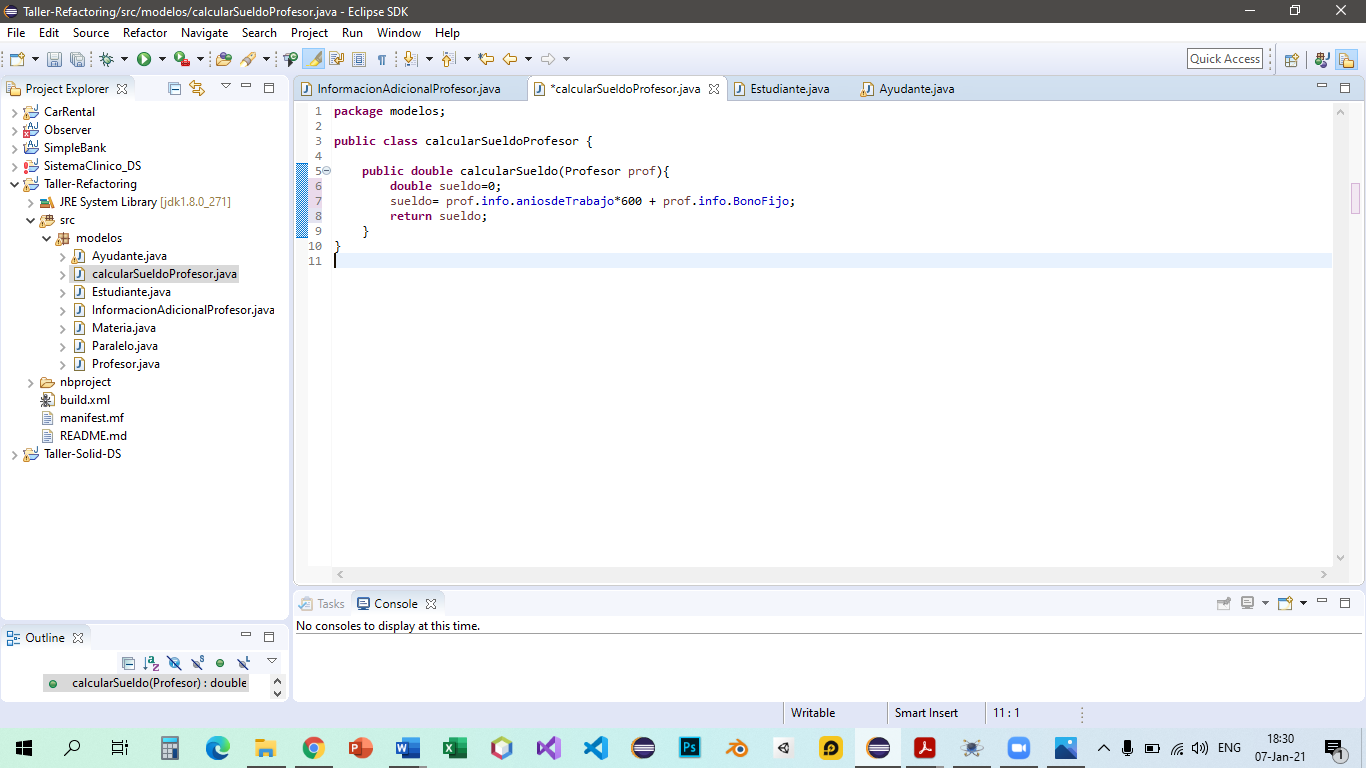
# Code Smell: Temporary Field

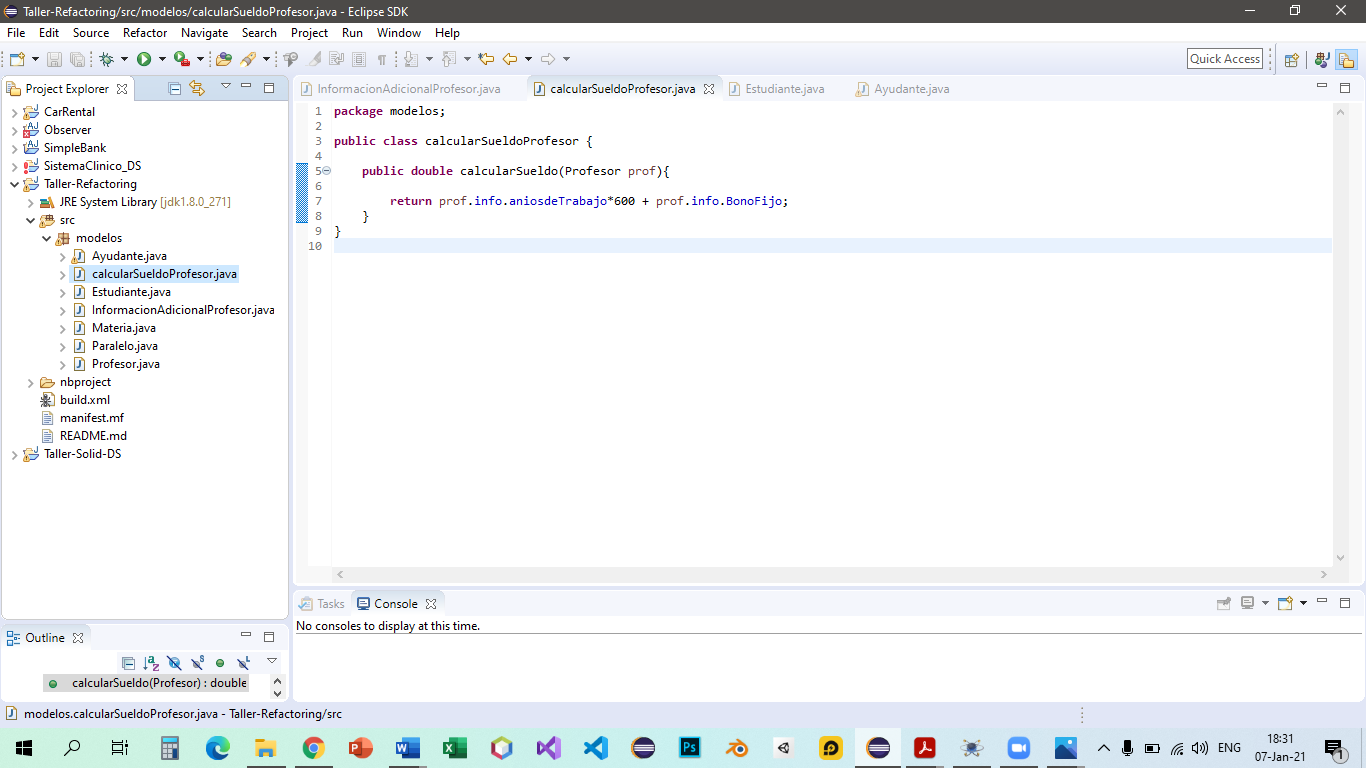
**Consecuencias:** Es una variable que solo se usa una vez usando recursos de manera innecesaria y aumenta líneas de código sin justificación.

**Técnica de Refactorización para utilizar:** Inline Temp.

Esta técnica permitiría obtener el resultado de la expresión de manera directa y sin desperdiciar recursos.

Código Anterior:



Código Nuevo.

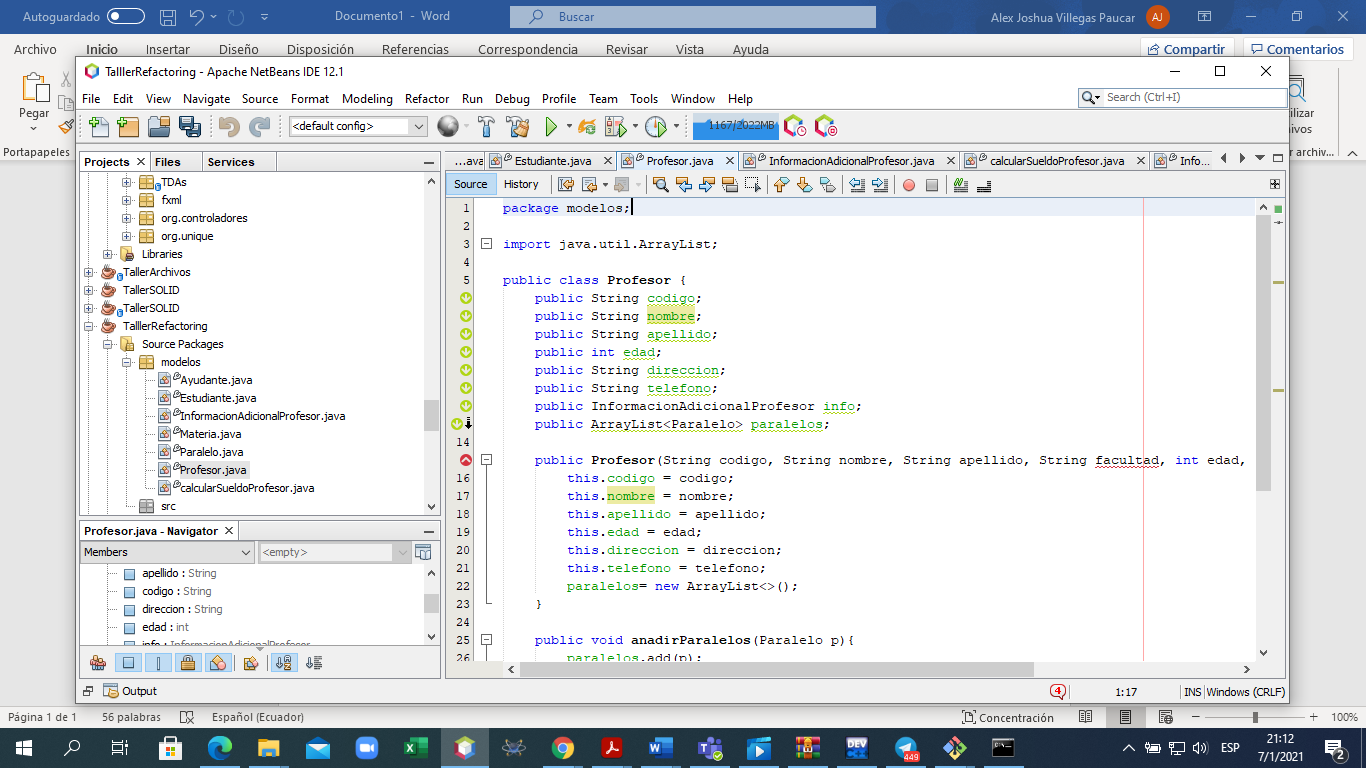
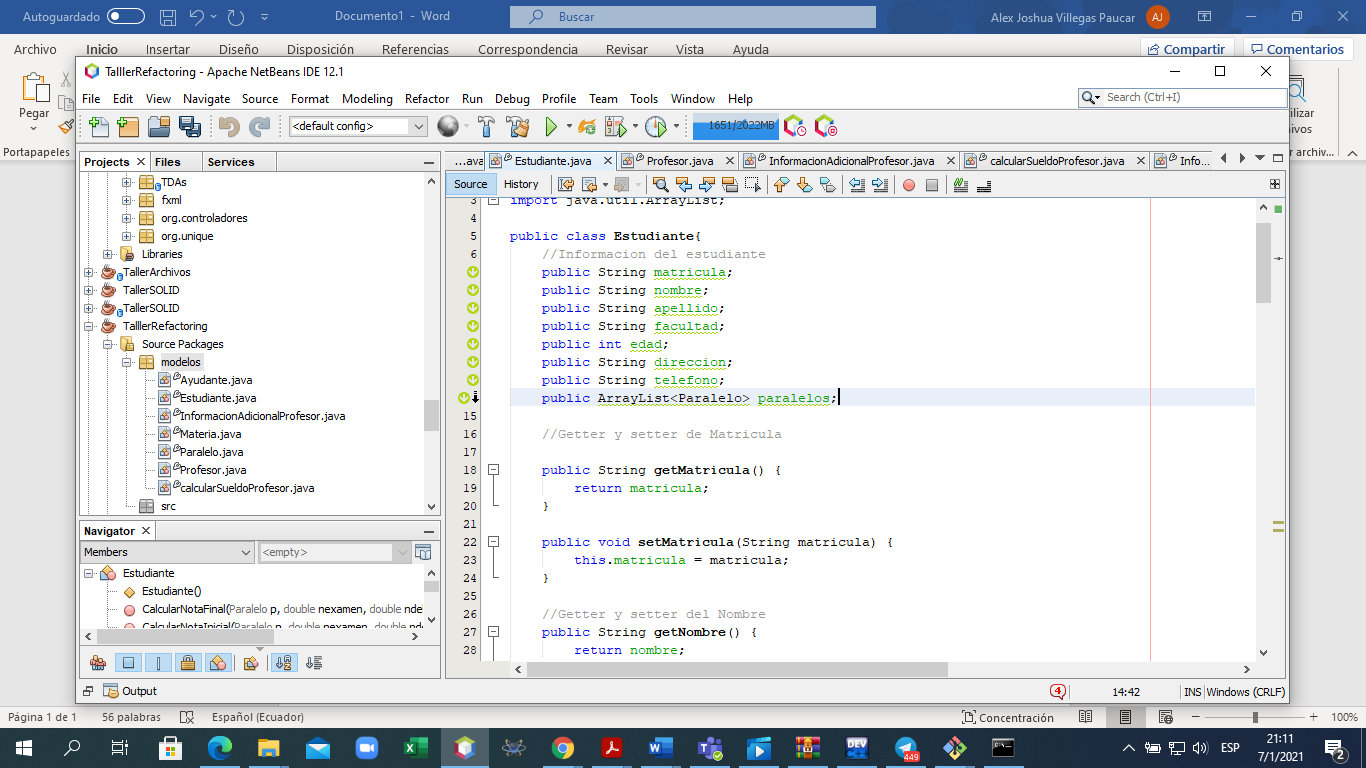
# Code Smell: Primitive Obsession

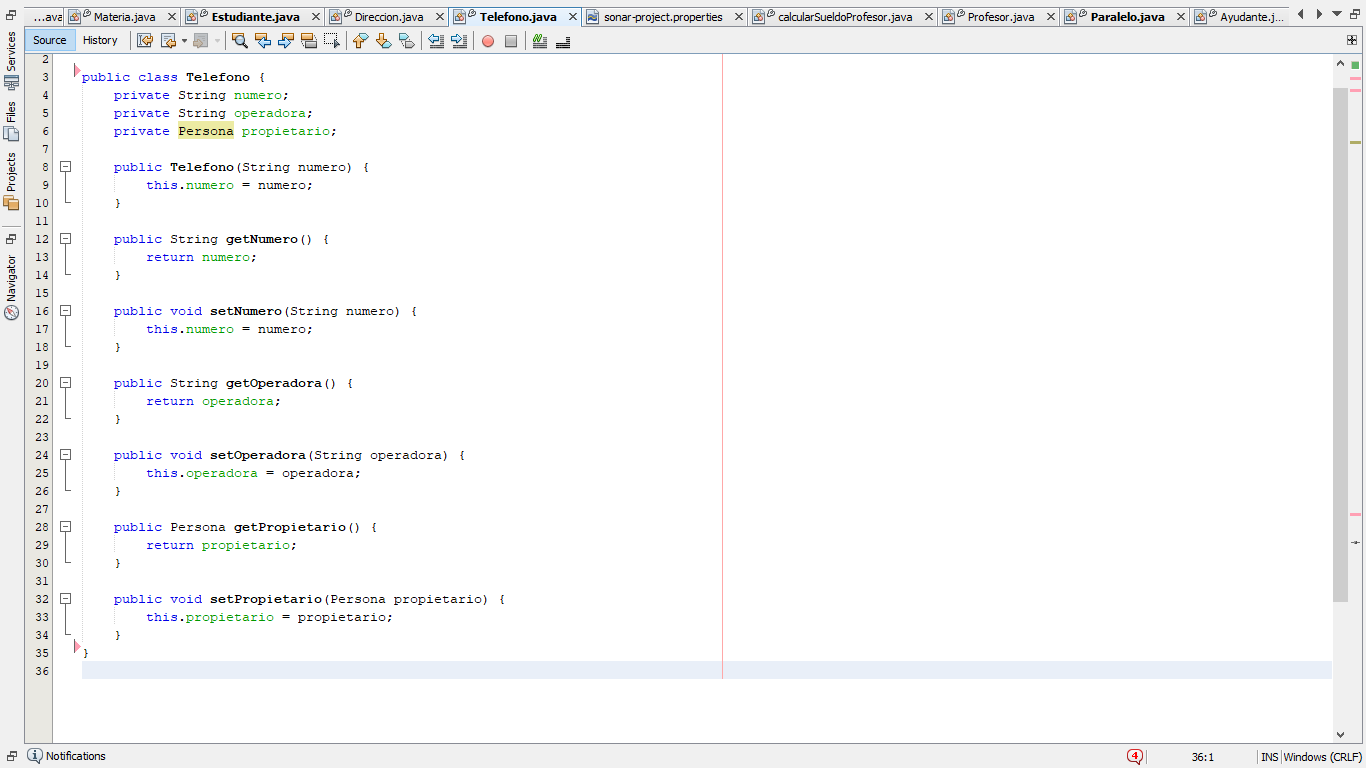
**Consecuencias:** La consecuencia de mantener a las variables dirección y teléfono como primitivas, es que no permite describir el comportamiento adecuado de esos atributos, pues el tipo primitivo no le brinda una adecuada descripción a su data.

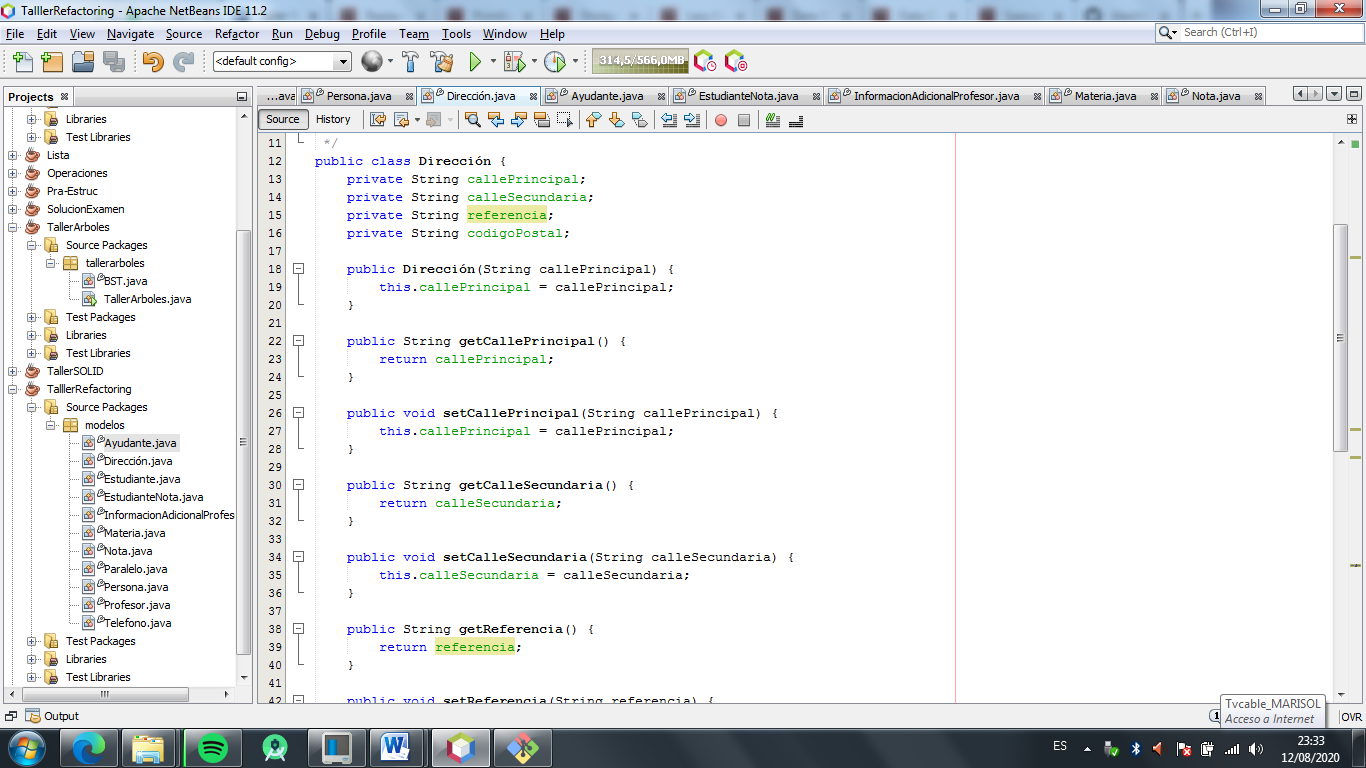
**Técnica de Refactorización para utilizar:** Replace Data Value with Object

Revisando ambos atributos que usa las clases Profesor y Estudiante, se decidió agregar un comportamiento descriptivo que estaría mejor ubicada en nuevas clases. Dado este caso se utilizaría la técnica de refactorización **Replace Data Value with Object:**

Como ambos atributos no cumplen con un comportamiento adecuado, se crean clases de con un nombre descriptivo y se agregan las funcionalidades descriptivas y adecuadas de cada una de las clases.

Código anterior:

Código actualizado:



# Code Smell: Lazy class

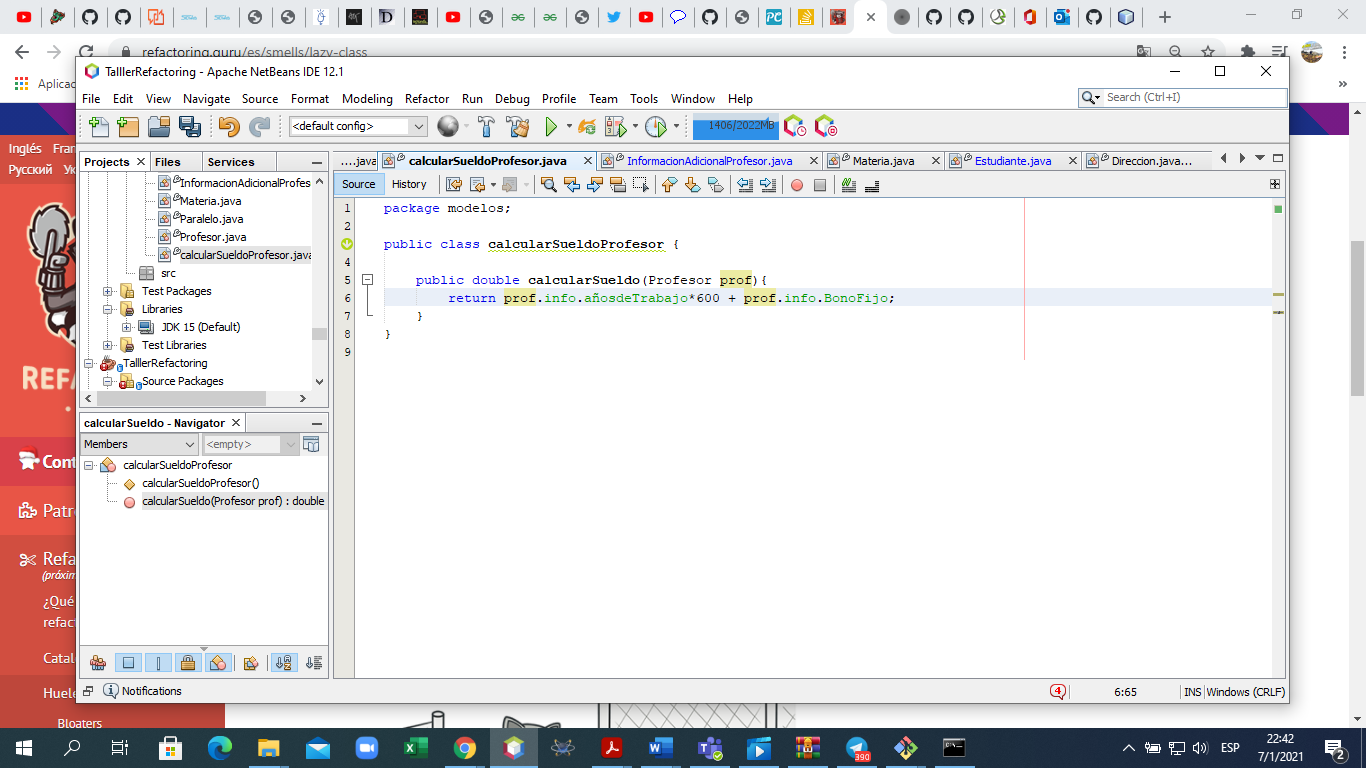
**Consecuencias:** Las consecuencias de tener una Lazy Class es que los costos de mantenimiento pueden elevarse, ya que habrá más código que leer, además que quitará tiempo.

**Técnica de Refactorización para utilizar:** Inline Class, Move Method

Inline class: Para solucionar el code smell se utilizará el método Inline Classlo cual nos permitirá unir una clase con otra, en este caso podremos eliminar la clase CalcularSueldoProfesor.

Move method: Este nos ayudara a mover el método que contenía la clase CalcularSueldoProfesor a la clase InformacionAdicionalProfesor**.**

Código anterior:



Código actualizado:

