**ESCUELA SUPERIOR POLITECNICA DEL LITORAL**

**FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION**

**Materia:**

DISEÑO DE SOFTWARE

**Tema:**

CODE SMELLS Y TECNICAS DE REFACTORIZACION

**Integrantes:**

* CORDOVA ANCHUNDIA LEA ALFONSINA
* LOPEZ VACA BRIGGITTE LIZBETH
* SEGURA MERCHAN RONNY HUGO

**Paralelo:**

3

**1S-2020**

**Contenido**

[**CODE SMELLS** 4](#_Toc48244922)

[**1.** **DUPLICATE CODE** 4](#_Toc48244923)

[1.1. Descripción 4](#_Toc48244924)

[1.2. Capturas del Code Smell 4](#_Toc48244925)

[1.3. Consecuencias 4](#_Toc48244926)

[1.4. Técnica(s) de refactorización: Subir Campo 5](#_Toc48244927)

[1.5. Capturas del código refactorizado 5](#_Toc48244928)

[**2.** **LONG PARAMETER LIST** 6](#_Toc48244929)

[2.1. Descripción 6](#_Toc48244930)

[2.2. Capturas del Code Smell 6](#_Toc48244931)

[2.3. Consecuencias 6](#_Toc48244932)

[2.4. Técnica(s) de refactorización 6](#_Toc48244933)

[2.5. Capturas del código refactorizado 7](#_Toc48244934)

[**3.** **SPECULATIVE GENERALITY** 8](#_Toc48244935)

[3.1. Descripción 8](#_Toc48244936)

[3.2. Capturas del Code Smell 8](#_Toc48244937)

[3.3. Consecuencias 8](#_Toc48244938)

[3.4. Técnica(s) de refactorización 8](#_Toc48244939)

[3.5. Capturas del código refactorizado 8](#_Toc48244940)

[**4.** **LARGE CLASS** 9](#_Toc48244941)

[4.1. Descripción 9](#_Toc48244942)

[4.2. Capturas del Code Smell 9](#_Toc48244943)

[4.3. Consecuencias 9](#_Toc48244944)

[4.4. Técnica(s) de refactorización: Extraer Clase 9](#_Toc48244945)

[4.5. Capturas del código refactorizado 9](#_Toc48244946)

[**5.** **Nombre CS3** 10](#_Toc48244947)

[5.1. Descripción 10](#_Toc48244948)

[5.2. Capturas del Code Smell 10](#_Toc48244949)

[5.3. Consecuencias 10](#_Toc48244950)

[5.4. Técnica(s) de refactorización 10](#_Toc48244951)

[5.5. Capturas del código refactorizado 10](#_Toc48244952)

[**6.** **Nombre CS4** 10](#_Toc48244953)

[6.1. Descripción 10](#_Toc48244954)

[6.2. Capturas del Code Smell 10](#_Toc48244955)

[6.3. Consecuencias 10](#_Toc48244956)

[6.4. Técnica(s) de refactorización 10](#_Toc48244957)

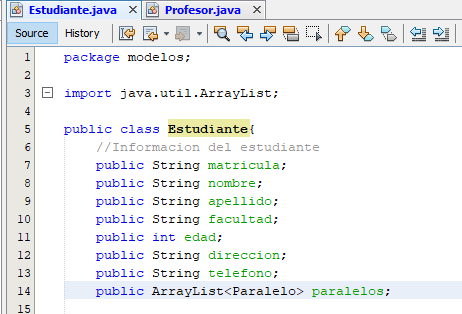
[6.5. Capturas del código refactorizado 10](#_Toc48244958)

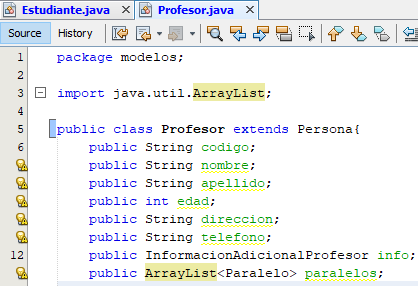
**CODE SMELLS**

1. **DUPLICATE CODE** 
   1. Descripción

Este code smells se da cuando dos fragmentos de código son exactamente iguales, tal como se daba en las clases Estudiantes y Profesor, los cuales compartían atributos como nombre, apellido, edad, dirección, teléfono y paralelos y estos atributos eran del mismo tipo.

* 1. Capturas del Code Smell





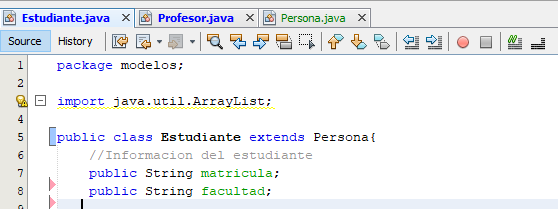
* 1. Consecuencias

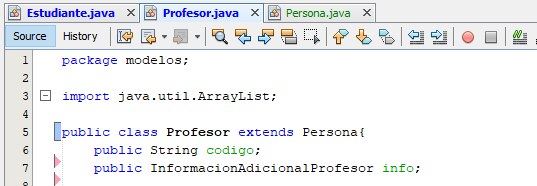
Las consecuencias es que se va a tener dos clases que comparten los mismos atributos lo que puede resultar en que en el futuro cuando necesitemos hacer algún cambio o mejora se nos va a hacer difícil, cambios que tendremos que realizar manualmente lo que significara en que se usaran más recurso como el tiempo.

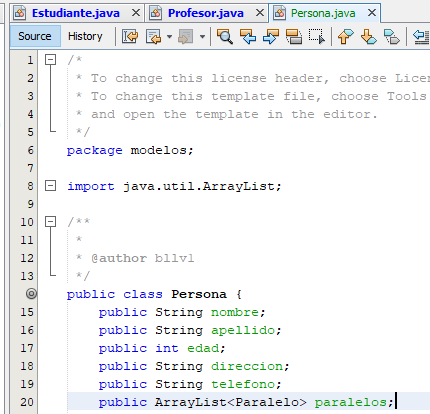
* 1. Técnica(s) de refactorización: Subir Campo

Se utilizo esta técnica para extraer los parámetros nombre, apellido, edad, dirección, teléfono y paralelos y se creó una clase padre llamada “Persona” con el fin de que las clases Estudiante y Profesor extiendan de esta clase y así solucionar el code smells de código duplicado.

* 1. Capturas del código refactorizado





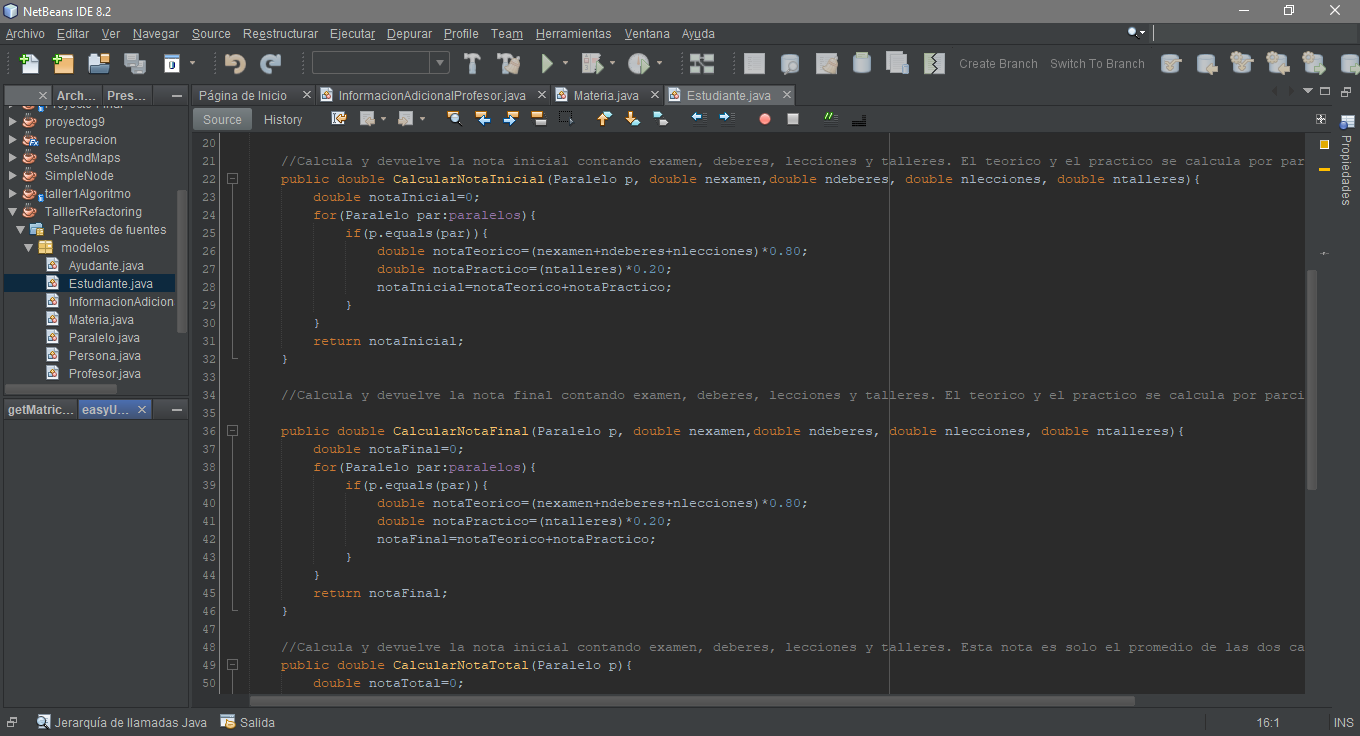


1. **LONG PARAMETER LIST**
   1. Descripción

Este olor implica que los métodos contengan más de cuatro parámetros, lo cual se hace que esta lista se vuelva difícil de usar o contradictoria a medida que se añadan.

Lo que se pretende con su solución es pasar lo necesario como parámetro en un método o utilizar los datos del objeto que contenga lo necesario.

* 1. Capturas del Code Smell



* 1. Consecuencias

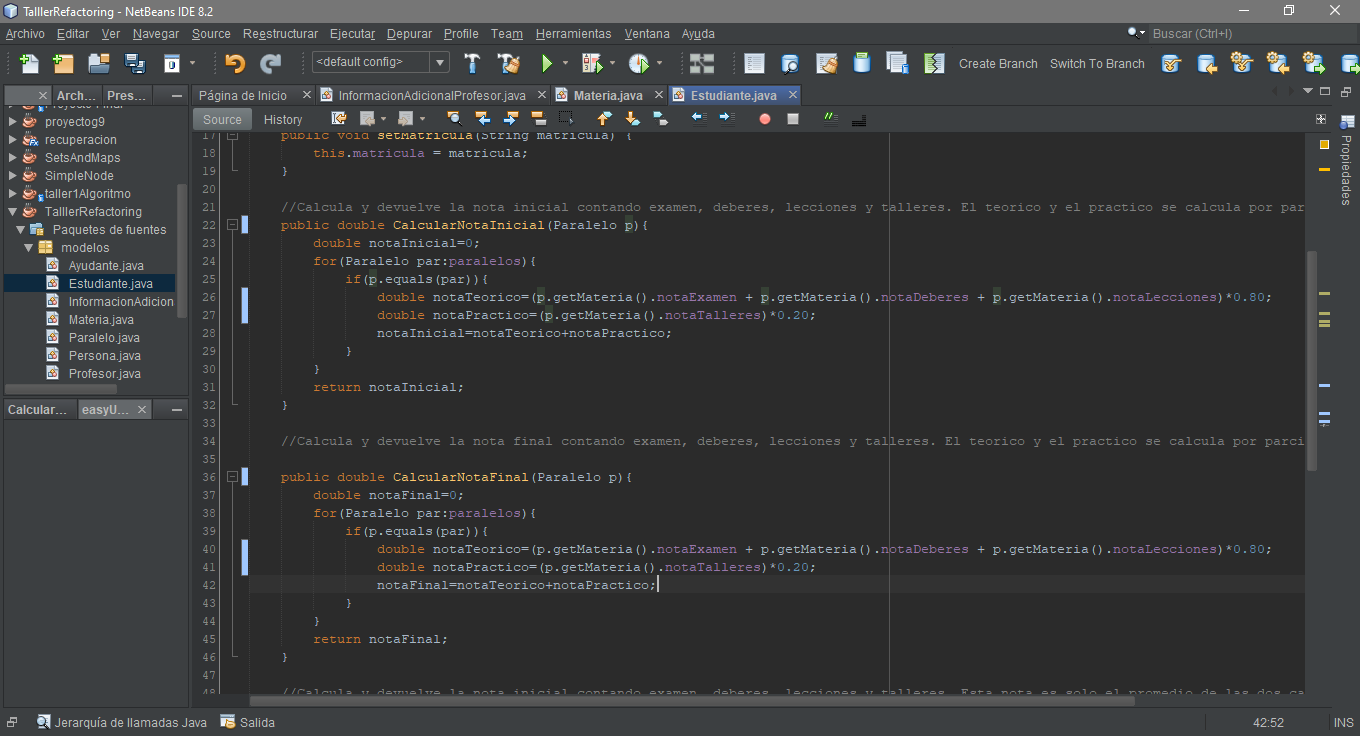
Este olor hace que el código no sea tan legible y su solución con refactorización implicaría que se genere código duplicado previamente invertido.

* 1. Técnica(s) de refactorización

Para la refactorización se aplicó **Perserver Whole Object** en lo cual consiste remplazar una la lista de parámetros que recibe por solo el objeto que maneje los datos necesarios.

También se aplicó Introduce **Parameter Object** donde los datos similares que son enviados como parámetros son remplazados por un objeto. Al aplicar esta refactorización se movieron los datos a otra clase por ende puede generar a una clase de datos, también como los métodos son dependientes de otra clase se los puede mover, para ello se usa **Move Method**.

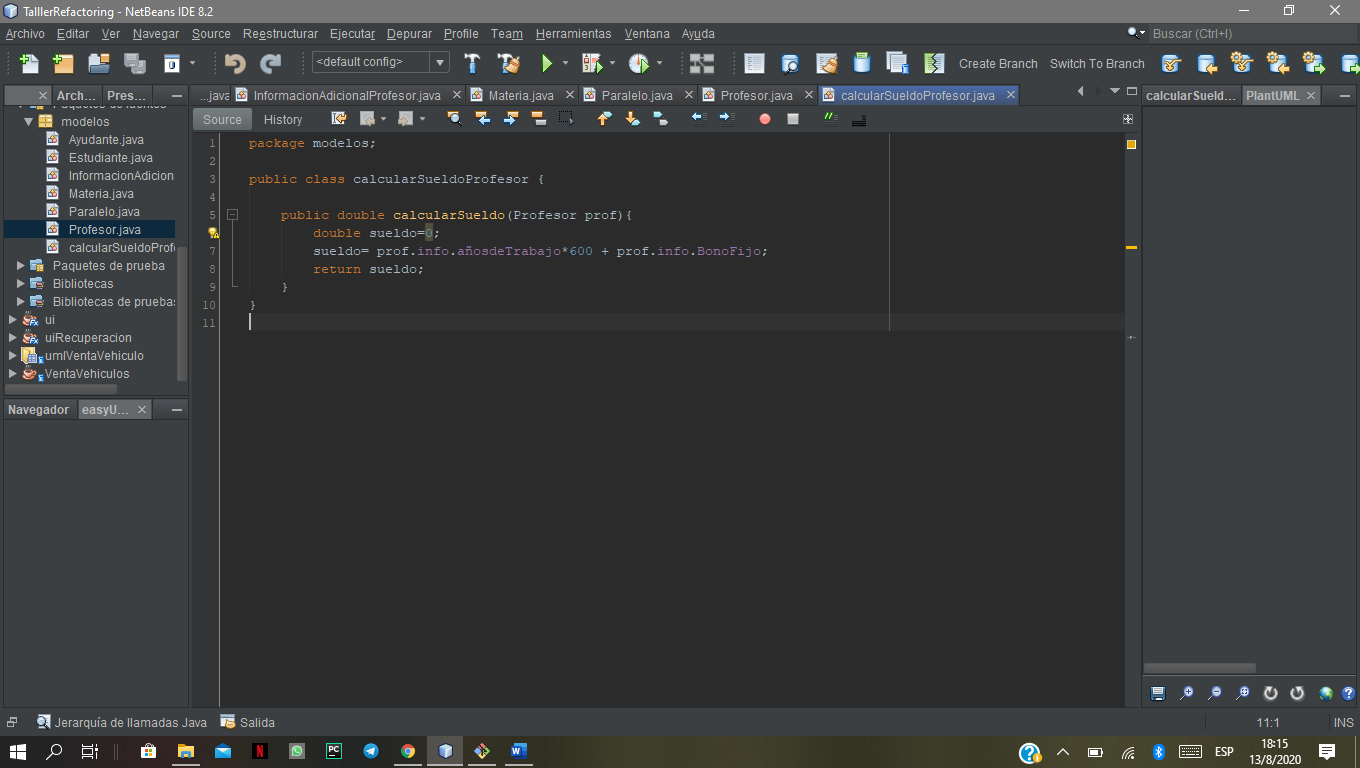
* 1. Capturas del código refactorizado



1. **SPECULATIVE GENERALITY**
   1. Descripción

Este olor trata cuando existen clases con características que se anticipan y que nunca se implementan, por ende, el código se hace difícil de entender. Este olor hace que la clase CalcularSueldoProfesor no se ha utilizado.

* 1. Capturas del Code Smell



* 1. Consecuencias

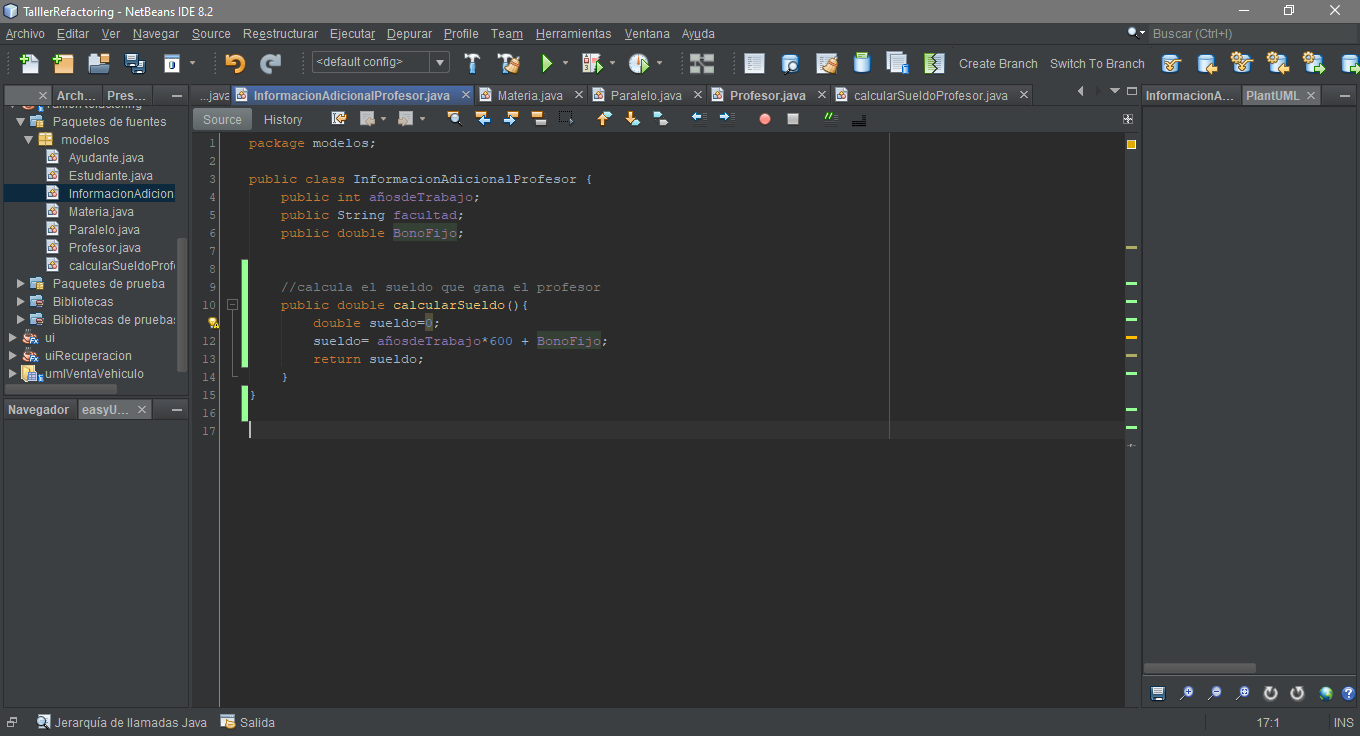
Si no se refactoriza el olor hará que el código sea más robusto y por ende su soporte sea complicado.

* 1. Técnica(s) de refactorización

Para la refactorización se aplicó **Inline Class**, para delegar la función de esta clase en otra, para que esta clase pueda ser eliminada y por ende tener un código más ligero.

**Inline Class** implica que una clase dentro del código no cumpla ninguna función por ende se deben mover todas las funciones a otra clase.

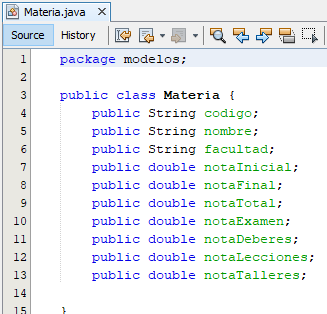
* 1. Capturas del código refactorizado



1. **LARGE CLASS**
   1. Descripción

Este code smell se da cuando una clase tiene muchas líneas de código, parámetros, métodos o muchos campos.

* 1. Capturas del Code Smell



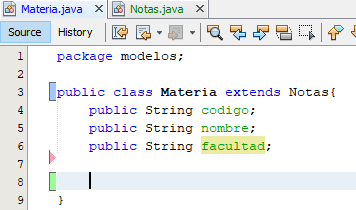
* 1. Consecuencias

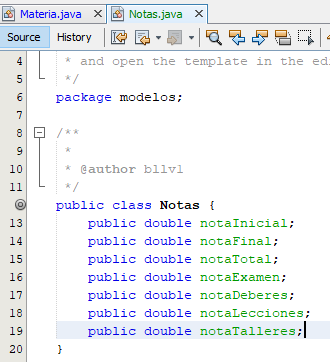
Las consecuencias de este code smells es que con el tiempo cuando el código necesita mantenimiento, entonces las clases crecen. Además, de que para los programadores es más fácil añadir una nueva característica en una clase existente a que crear una nueva clase para una nueva característica.

* 1. Técnica(s) de refactorización: Extraer Clase

Para resolver este code smells se utilizó la técnica de refactorización de extraer clase donde se creo la clase Notas donde se trasladaron los parámetros relevantes para esta clase y que no son tan relevantes para la clase Materia

* 1. Capturas del código refactorizado

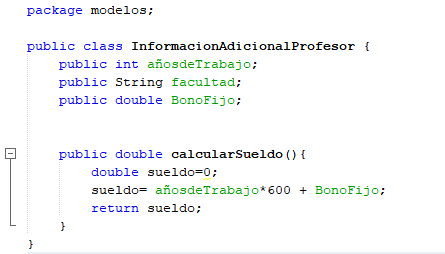




1. **LAZY CLASS**
   1. Descripción

Este code Smell tiene que ver con las clases que están demás es decir que no tienen ninguna función y no son usadas en ningún sentido.

* 1. Capturas del Code Smell



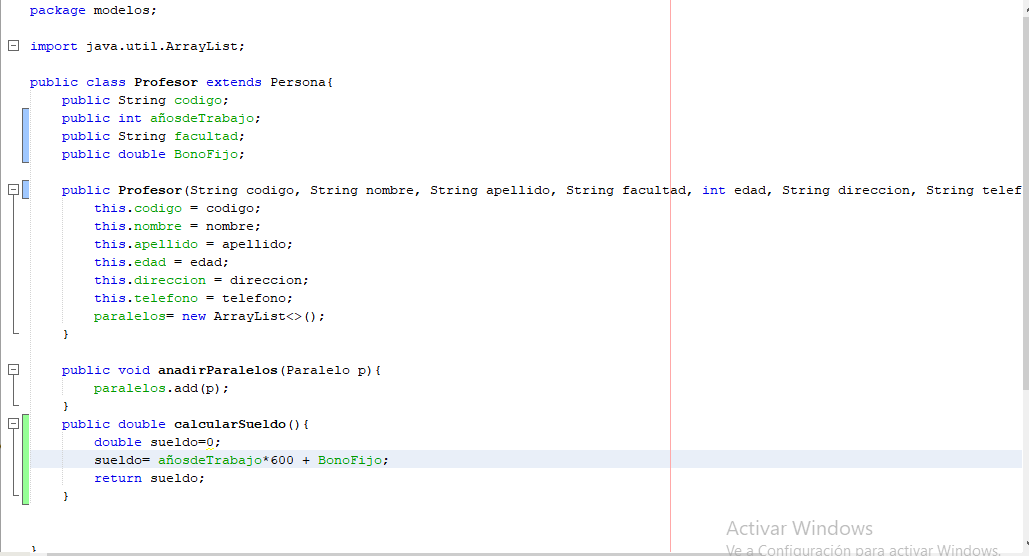
* 1. Consecuencias

Las consecuencias que vendría a tener este code Smell es que a medida que se le dan mantenimiento o se va actualizando el programa se tiene a crear mas clases. Entonces el tener estas clases que no tienen ninguna función hace que el programa se vea cargado y para el programador puede llegar a ser tedioso.

* 1. Técnica(s) de refactorización

La técnica que utilizamos para resolver este smell es la de inline class la cual explica que si se tiene una clase que prácticamente no se usa y no hace nada la solución que indica es que el contenido de la clase inútil se lo coloque dentro de esta clase para proceder a eliminarla.

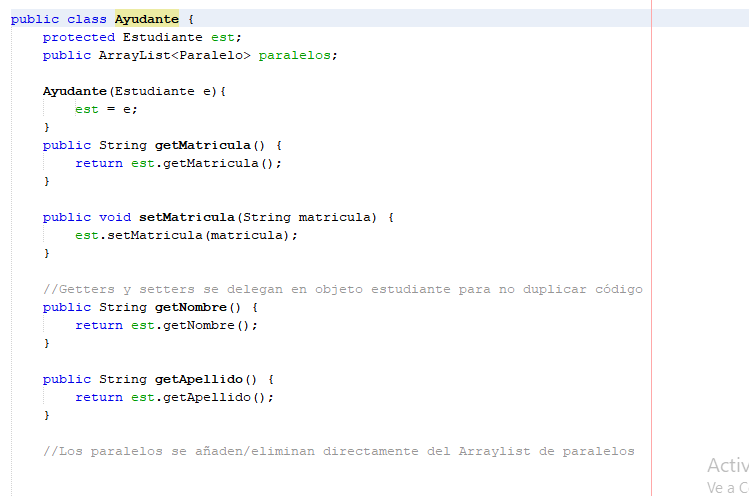
* 1. Capturas del código refactorizado



1. **INAPPROPRIATE INTIMACY**
   1. Descripción

Este code Smell tiene que ver con el tipo de clases que utiliza los campos o métodos perteneciente de otra clase.

* 1. Capturas del Code Smell

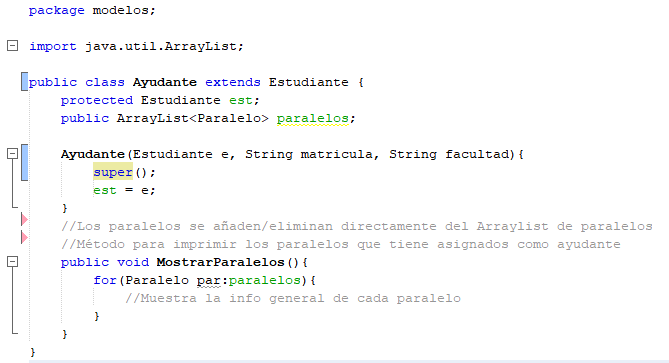


* 1. Consecuencias

Las consecuencias de este code Smell pueden generar desorganización dentro del código puesto que existe muchas dentro de las clases además que la reutilización de códigos conllevaría a generar errores y tener que cambiar ciertas cosas cada que se quiera realizar una implementación.

* 1. Técnica(s) de refactorización

Se utiliza la tecnica de Replace Delegation with Inheritance el cual consiste en hacer que la clase se extienda de otra clase en pocas palabras haciendo herencia. De este modo la dependencia de las clases será casi nula.

* 1. Capturas del código refactorizado