Taller 09

Code Smells

Estudiantes:

* Karla Durán Oscuez
* Bryan Plaza Anchundia
* Betsy Nazareno Aguiño

Profesor:

Msc. David Jurado Mosquera

Curso:

Diseño de Software

Contenido

[Code Smell: Long Class 4](#_Toc48246337)

[Código inicial 4](#_Toc48246338)

[Consecuencias y técnicas de refactorización 4](#_Toc48246339)

[Código modificado 5](#_Toc48246340)

[Code Smell: Long Parameter List 6](#_Toc48246341)

[Código inicial / code smell 6](#_Toc48246342)

[Consecuencias y técnicas de refactorización 6](#_Toc48246343)

[Código modificado 7](#_Toc48246344)

[Code Smell: Feature Envy 8](#_Toc48246345)

[Código inicial / code smell 8](#_Toc48246346)

[Consecuencias y técnicas de refactorización 8](#_Toc48246347)

[Código modificado 8](#_Toc48246348)

[Code Smell: Data class 9](#_Toc48246349)

[Código inicial / code smell 9](#_Toc48246350)

[Consecuencias y técnicas de refactorización 9](#_Toc48246351)

[Código modificado 9](#_Toc48246352)

[Code Smell: Duplicate code 10](#_Toc48246353)

[Código inicial / code smell 10](#_Toc48246354)

[Consecuencias y técnicas de refactorización 10](#_Toc48246355)

[Código modificado 11](#_Toc48246356)

[Code Smell: Inapropiate intimacy 11](#_Toc48246357)

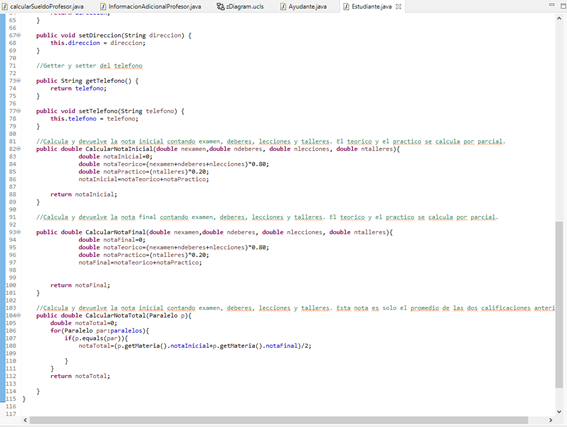
[Código inicial / code smell 11](#_Toc48246358)

[Consecuencias y técnicas de refactorización 12](#_Toc48246359)

[Código modificado 12](#_Toc48246360)

# Code Smell: Long Class

## Código inicial

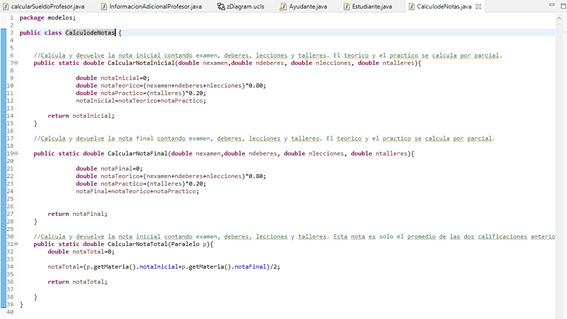


## Consecuencias y técnicas de refactorización

La clase Estudiante del proyecto presenta el code smell Long Class, puesto que contiene muchos métodos, concluyendo en muchas líneas de código. Se ha optado por extraer una clase, de nombre CalculodeNotas esto porque los métodos CalcularNotaInicial, CalcularNotaFinal y CalcularNotaTotal no deberían pertenecer a la clase Estudiante, hay una independencia, además de que la clase se libera de responsabilidades que no le corresponden, cumpliendo el principio SOLID de Single Responsability.

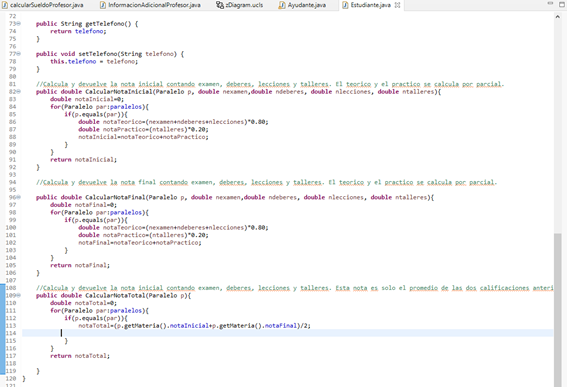
## Código modificado





# Code Smell: Long Parameter List

## Código inicial / code smell

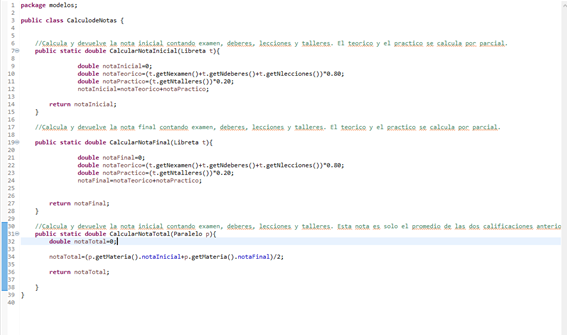


## Consecuencias y técnicas de refactorización

El code smell Long Parameter List en el presente proyecto tiene lugar en la clase CalculodeNotas clase creada de extraer métodos de la clase Estudiante, en específico en los métodos CalcularNotaInicial y CalcularNotaFinal. Por definición este code smell implica tener más de tres o cuatro parámetros para un determinado método, en los mencionados anteriormente hay cinco parámetros.

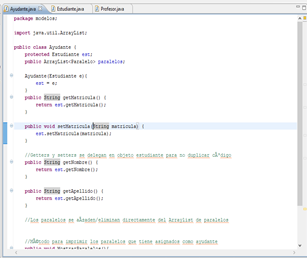
Haciendo un análisis el parámetro Paralelo puede ser extraído dado que este parámetro solo sirve para verificar si el paralelo existe en la lista del estudiante, esto no tendría mucho sentido, porque no se hace modificación alguna con respecto a algún campo del estudiante, por lo que esta lógica puede ser eliminada. Además, los otros parámetros como nexámen, ndeberes, nlecciones y ntalleres pueden ser reemplazados por un objeto, para esto se crea una clase que encapsula estos parámetros llamada Libreta, solucionando así el code smell con el método “Introduce Parameter Object”.

## Código modificado



# Code Smell: Feature Envy

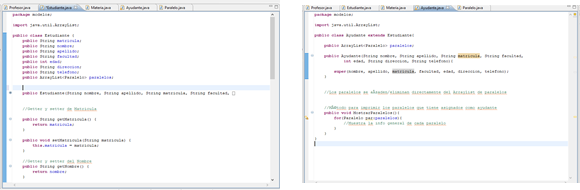
## Código inicial / code smell



## Consecuencias y técnicas de refactorización

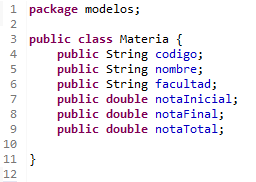
En este código se identificó el code Smell Feature Envy ya que la clase Ayudante usa de manera excesiva los métodos de la clase Estudiante. Cada uno de los métodos implementados en esta clase delegan el trabajo a los métodos de Estudiante. En vista de que estas clases comparten atributos y comportamientos similares, se puede resolver este codeSmell haciendo uso del polimorfismo para evitar esta delegación de comportamiento y más bien heredarlos de una clase padre, esta técnica de refactorización es conocida como Replace Delegation with Inheritance.

## Código modificado



# Code Smell: Data class

## Código inicial / code smell



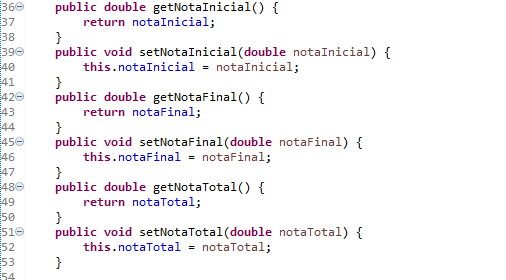
## Consecuencias y técnicas de refactorización

En la clase Materia se puede observar que sólo contiene atributos y no funcionalidades extras, lo cual hace que la clase solo sirva como reserva de data que otra clase podría utilizar (como es en el de calcularNota). Aquí no se está aprovechando la usabilidad que tiene una clase como tal, y es por eso que posee el code smell de Data Class. Las consecuencias de esto es que la clase se vuelve muy “estática” y se hace un poco más difícil entender luego el código.

Para corregir este code smell se utilizará la técnica de refactorización Encapsulated Field, el cual a los atributos públicos se los hará privados y además se le añadirá a la clase su respectivo constructor más los getters y setters.

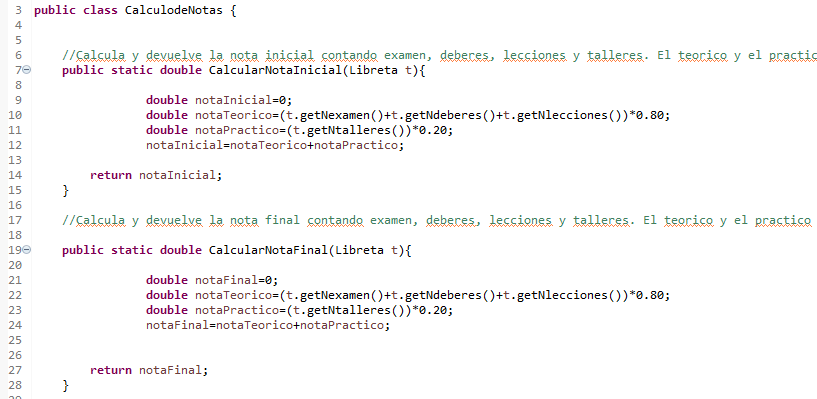
## Código modificado





# Code Smell: Duplicate code

## Código inicial / code smell

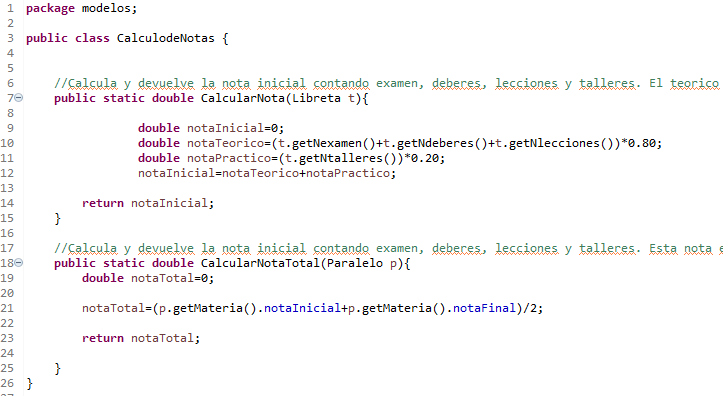


## Consecuencias y técnicas de refactorización

Luego de haber trasladado los métodos CalcularNotaInicial y CalcularNotaFinal de la clase Estudiante a la clase CalculodeNotas, se puede notar que ambos métodos a pesar de tener nombres diferentes, ambos poseen igual funcionalidad, es decir, poseen el mismo código. A este code smell se le llama código duplicado, y las consecuencias de tenerlo es que hace que la clase sea más extensa, lo que dificulta su legibilidad y que esté propenso a tener más errores y fallas en seguridad.

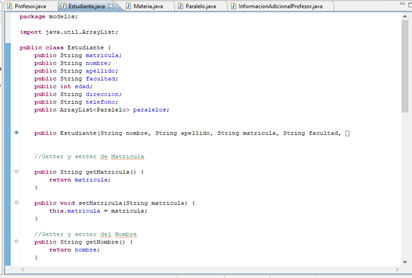
Para corregir este code smell se aplicará la técnica de refactorización Extract Method, donde se implementará un nuevo método llamado CalcularNota que tenga el mismo código que de ambos métodos duplicados, y posteriormente estos serán suprimidos.

## Código modificado



# Code Smell: Inapropiate intimacy

## Código inicial / code smell



## Consecuencias y técnicas de refactorización

En la clase Estudiante se puede apreciar que todos los atributos contenidos tienen el modificador de acceso público, lo cual permite que sean accedidos desde clases externas, esto es reconocido como el code Smell Inapropiate intimicy, para solucionarlo se cambiaran los modificadores de acceso a protected para que las únicas que tengan acceso a esta información sean las clases derivadas, esta encapsulación de atributos es conocido como una técnica de refactorización llamada: Encapsulated field.

## Código modificado

