7-1-2021

Taller refactoring

Contenido

[**Speculative Generality** 1](#_Toc60953653)

[**Duplicate Code** 2](#_Toc60953654)

[**Comments** 3](#_Toc60953655)

[**Data class** 4](#_Toc60953656)

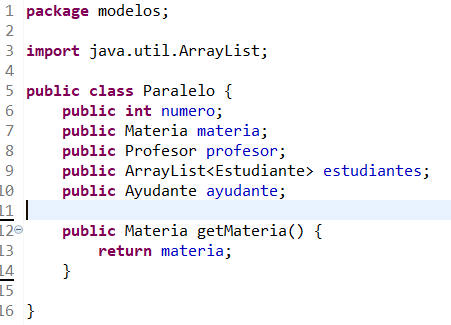
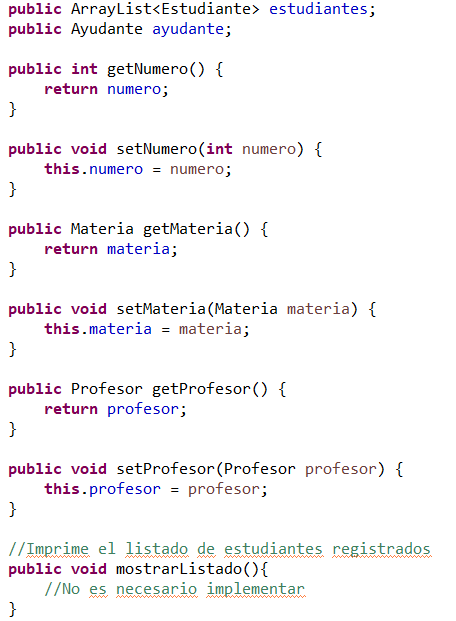
[**Long Parameter List** 5](#_Toc60953657)

[**Inappropriate intimacy** 6](#_Toc60953658)

Integrante

GILSON PONCE BRIONES

# **Speculative Generality**

****

**solución**

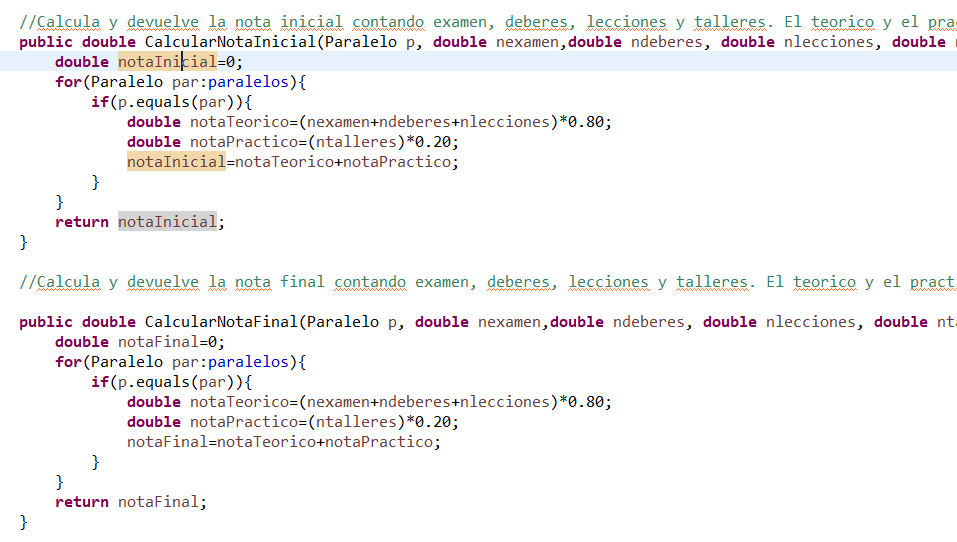
**Consecuencias**

Hay métodos que no son utilizados por ciertas clases en todo el proyecto, por lo que el código se puede volver difícil de entender y admitir.

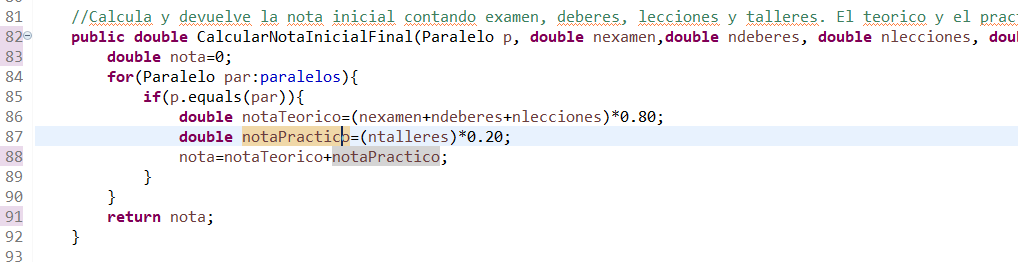
**Técnicas de refactorización que se pueden usar**

* Collapse Hierarchy
* Inline Class
* Remove Parameter
* O simplemente eliminar

# **Duplicate Code**

****

**Solución**

****

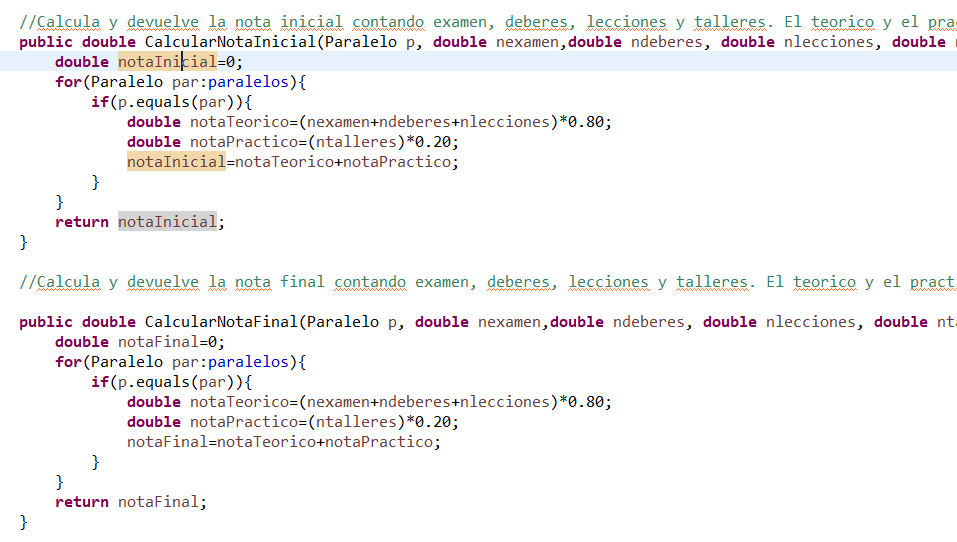
**Consecuencias**

Como vemos en las imágenes estos dos fragmentos de códigos parecen casi idénticos, esto puede suceder intencionalmente. Esto hace que la clase sea un poco más larga.

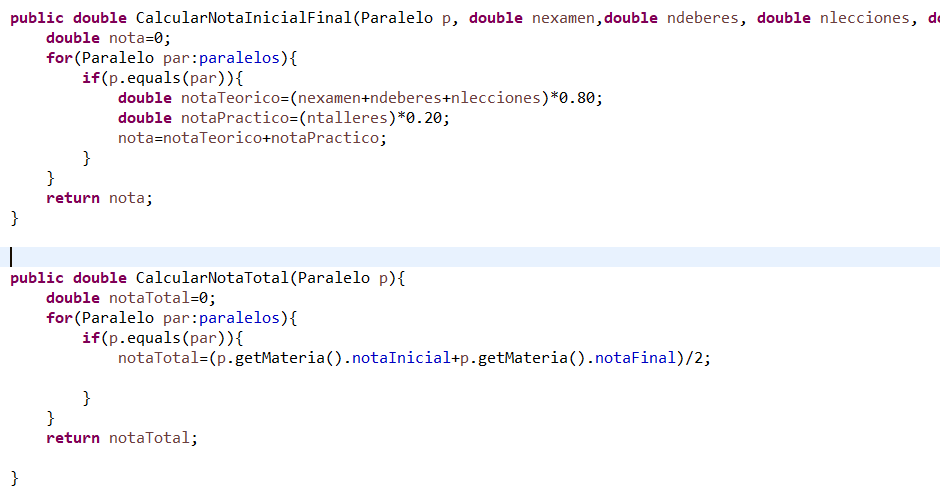
**Técnicas de refactorización que se pueden usar**

* Extract Method
* Subtitute Algorithm

# **Comments**

****

**Solución**

****

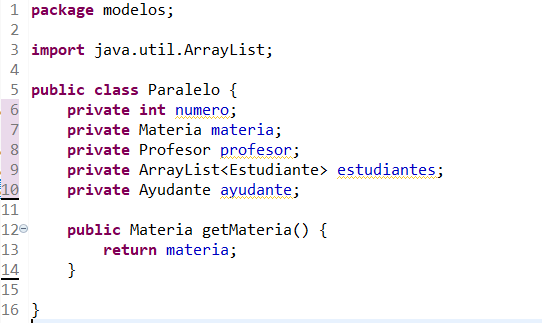
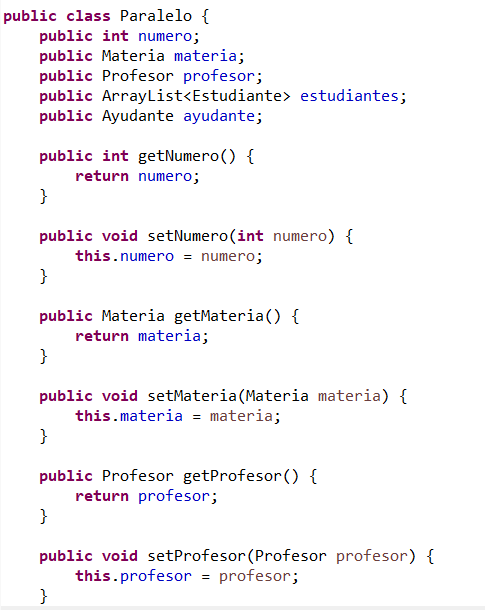
**Consecuencias**

Hay varias partes del proyecto donde se llenan de comentarios explícitos, ya que el nombre de los métodos da a conocer lo que hace; por lo que no se necesitan poner. Esto molesta mucho al momento de leer el código.

**Técnicas de refactorización que se pueden usar**

* Rename Method
* Introduce Assertion

# **Data class**

****

**solución**

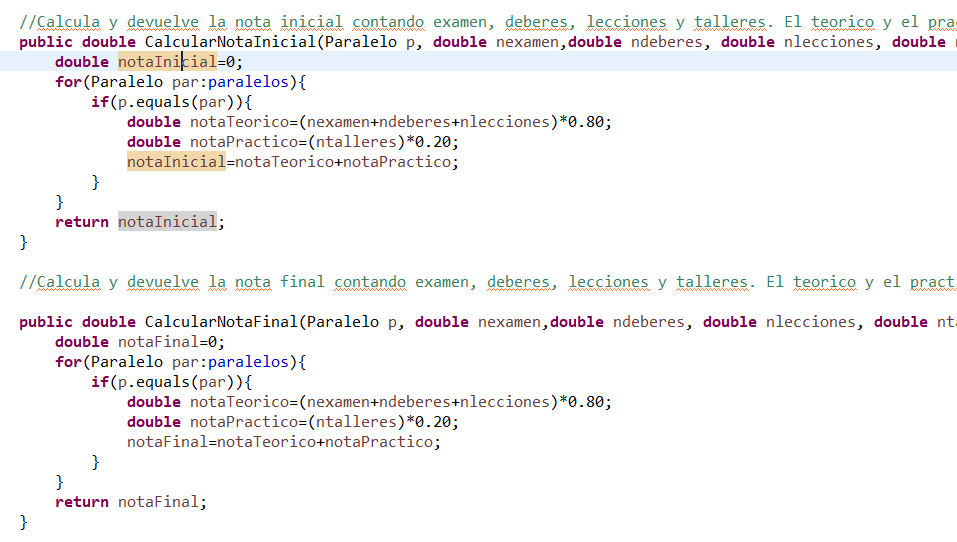
**Consecuencia**

Esta clase solo contiene campos y métodos crudos para obtener acceso a ellos. Sería mejor aprovecharlos para definir un comportamiento u operación con sus datos; mantenerlo no aprovecharíamos POO, ya que tiene atributos públicos. Además, incumpliríamos con los principios SOLID.

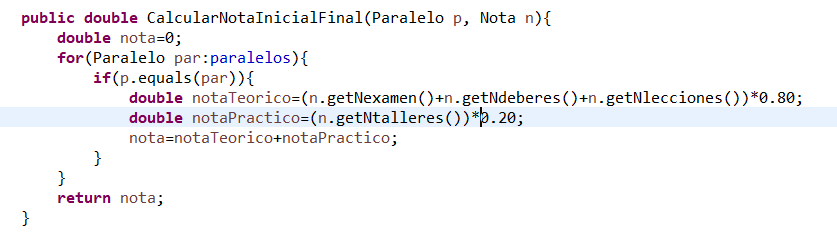
**Técnicas de refactorización que se pueden usar**

* Encapsulate field
* Encapsulate Collection

# **Long Parameter List**

****

**Solución**

****

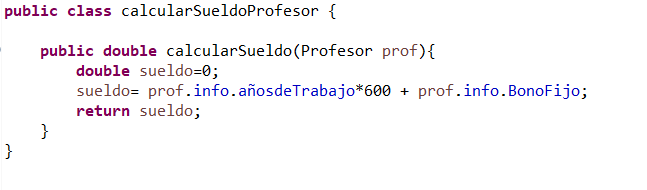
**Consecuencia**

Más de tres o cuatro parámetros para un método existe en este proyecto, en lugar de esto puede ser reemplazados por un objeto. En este caso hace que la lectura del código no se la mejor para su entendimiento, ya que abarca una línea extensa y no deja leer con fluidez.

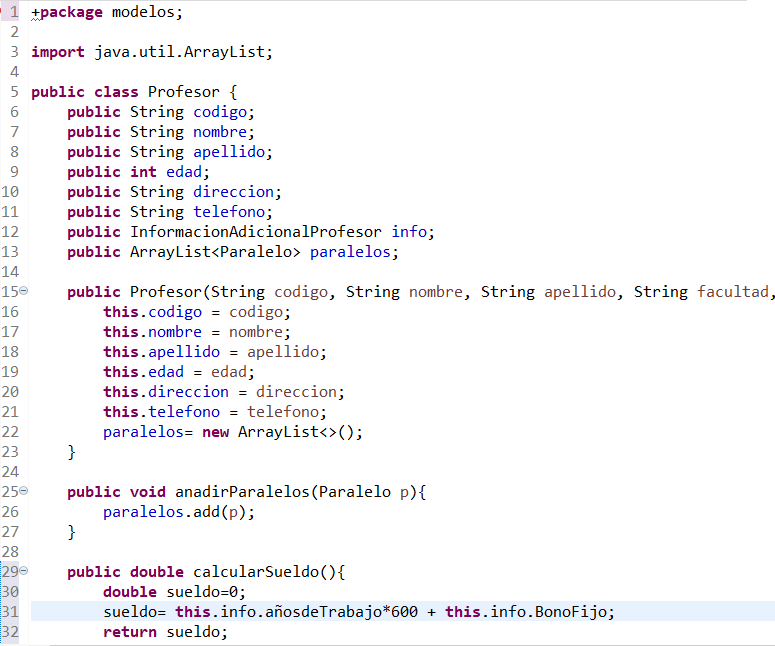
**Técnicas de refactorización que se pueden usar**

* Preserve Whole Object
* Indroduce Parameter Object

# **Inappropriate intimacy**

****

**Solución**

****

**Consecuencia**

Esta clase utiliza los campos de otra clase, por lo que siguiendo la buenas practicas; una buena clase debe saber lo menos que se pueden hacer. Teniendo el código así marea un poco al leer en la clase que lo utiliza y no se está aprovechando el paradigma POO; encapsulamiento.

**Técnicas de refactorización que se pueden usar**

* Extract class
* Hide delegate
* Metodo Move