Reporte Factoring

Integrantes:

- Andrea Soriano

- Bryan Puchaicela

- Kenny Yépez

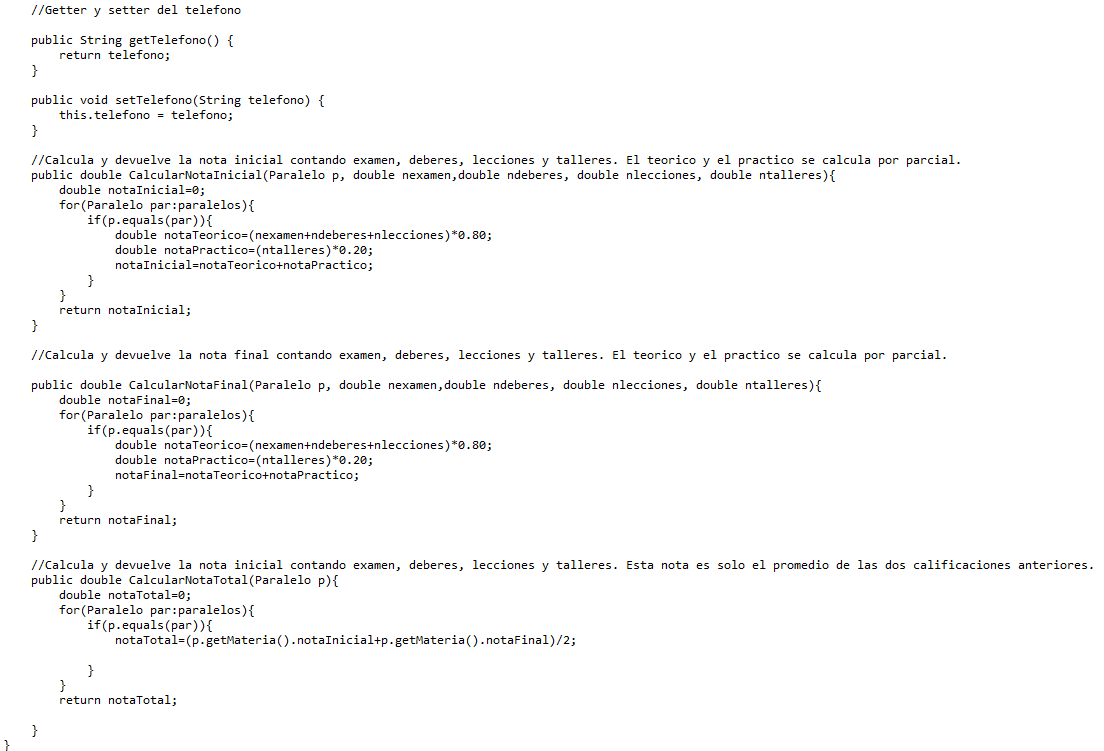
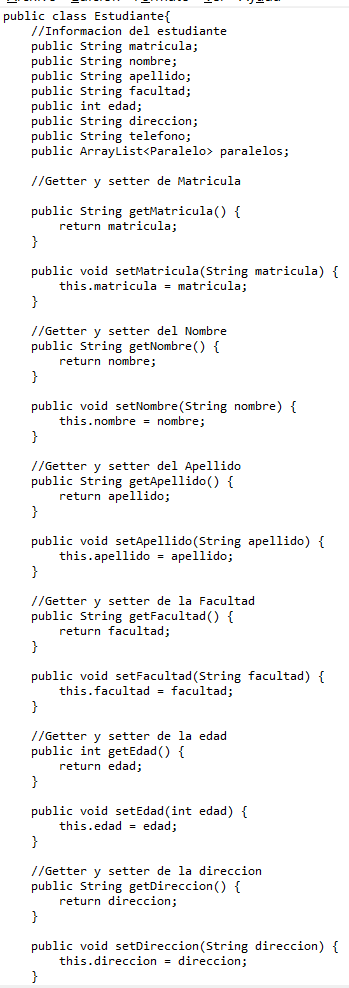
# Índice:

1. [Code Smells](#CodeSmells)
   1. [Large Class](#LargeClass)
   2. [Long Parameter List](#LongParameterList)
   3. [Data Clumps](#DataClumps)
   4. [Temporary Field](#TemporaryFields)
   5. [Alternative Classes with Different Interfaces](#AlternativeClasesWithDI)
   6. [Data Class](#DataClass)
   7. [Feature Envy](#FeatureEnvy)
   8. [Lazy Class](#LazyClass)
   9. [Inappropiate Intimacy](#InappropiateIntimacy)
   10. [Duplicated Code](#DuplicatedCode)

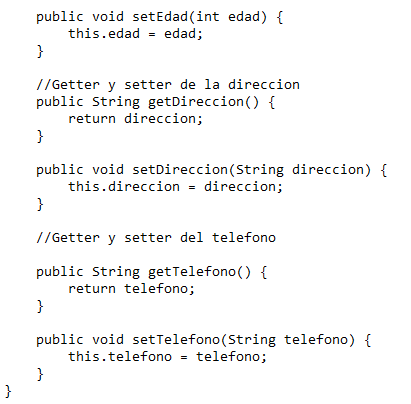
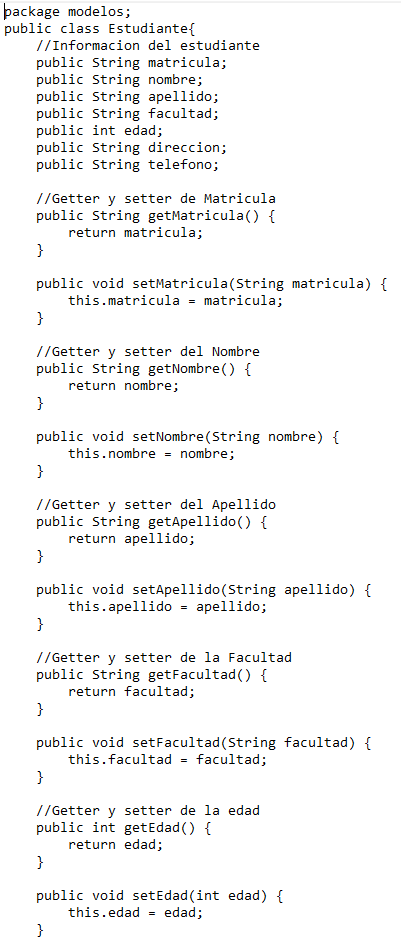
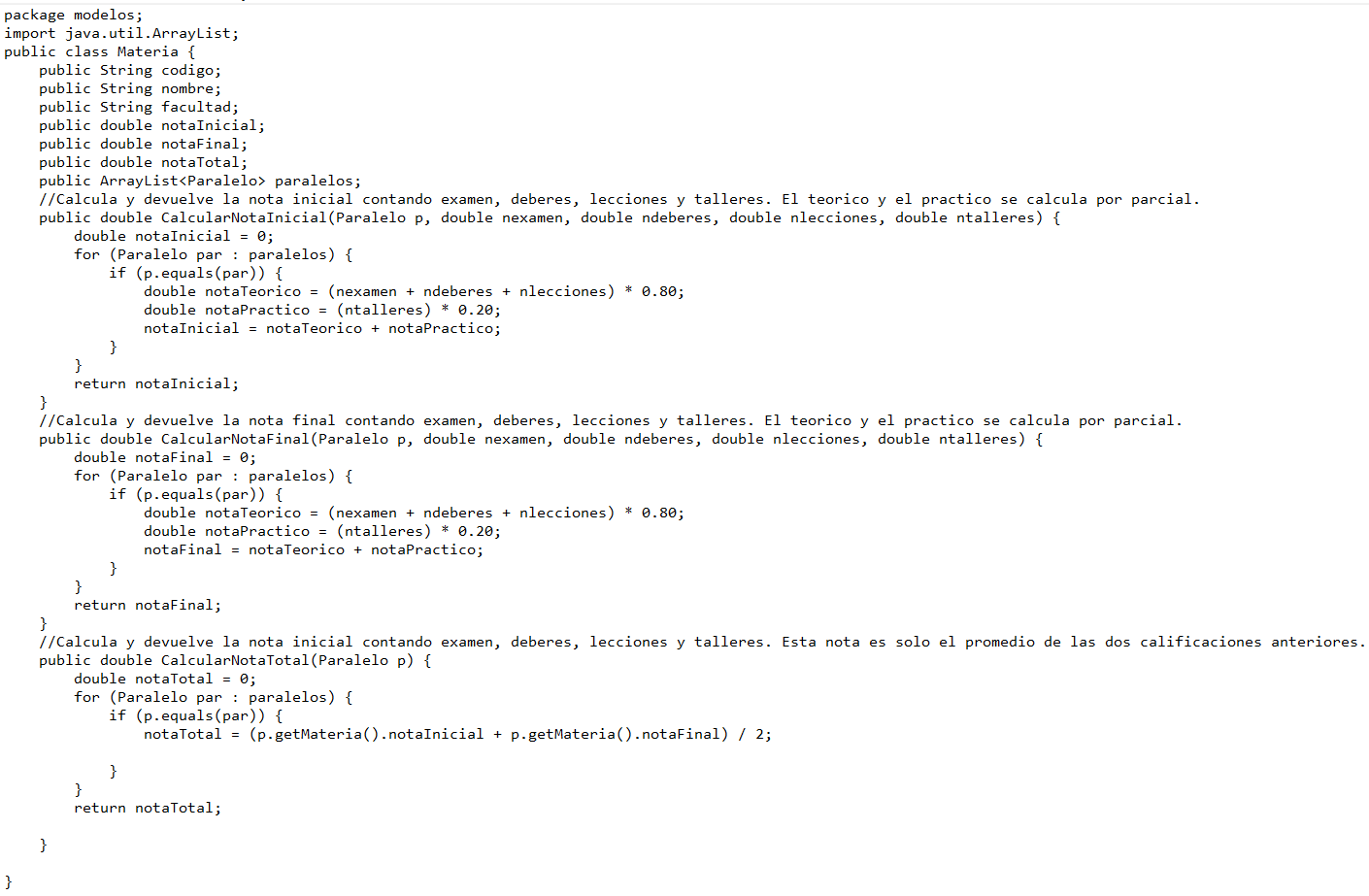
Code Smells

**Large Class**

**Código Original:**



**Código modificado:**



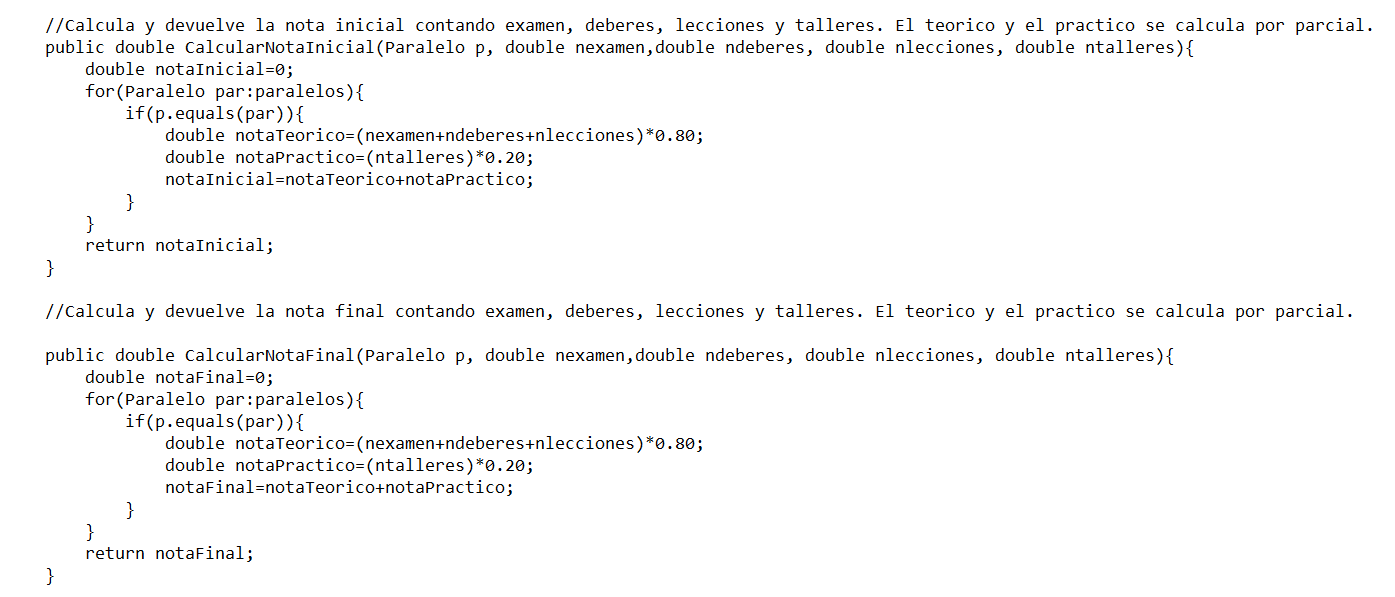
**Consecuencias:**

Dejando la clase igual, haría que los desarrolladores tengan que recordar demasiados atributos y métodos, además, es más probable que exista duplicación de código y funcionalidad.

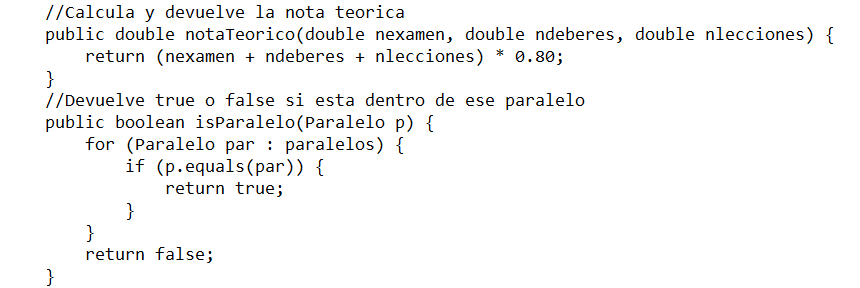
**Técnicas de refactorización:** Extract Class

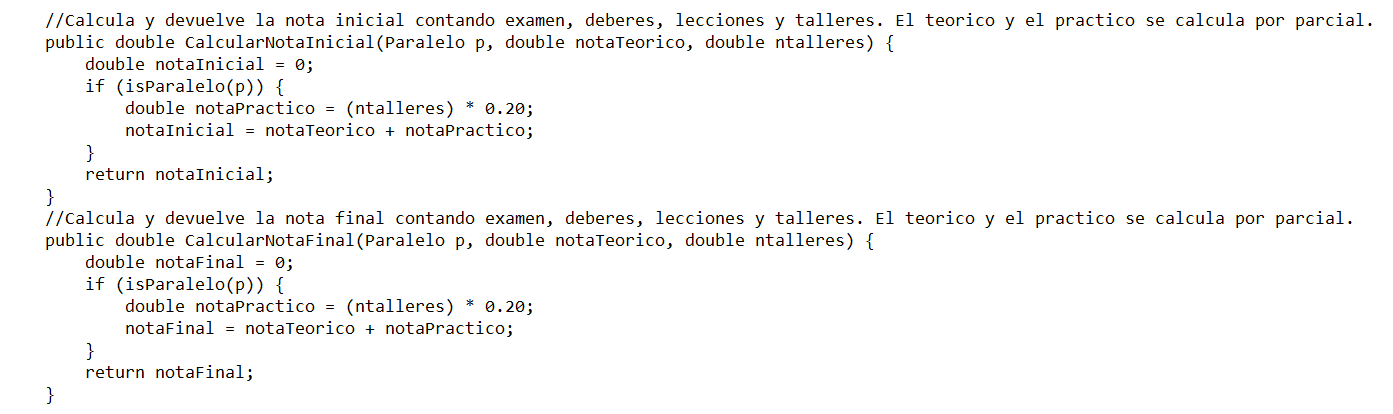
**Long Parameter List**

**Código original:**



**Código modificado:**





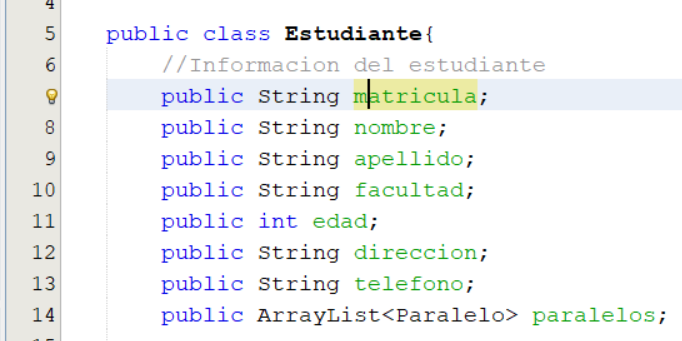
**Consecuencias:**

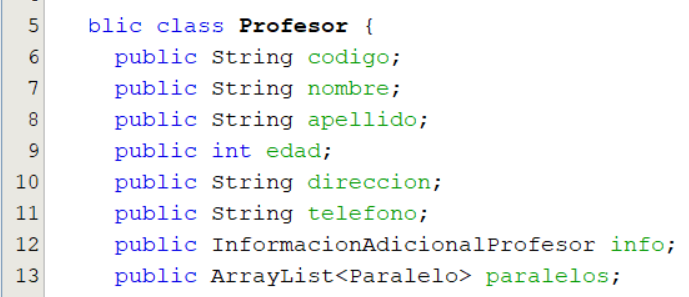
Manteniendo el código así será más difícil de leer y más largo, además existiría posibilidad de que haya código duplicado.

**Técnicas de refactorización:** Replace Parameter with Method Call

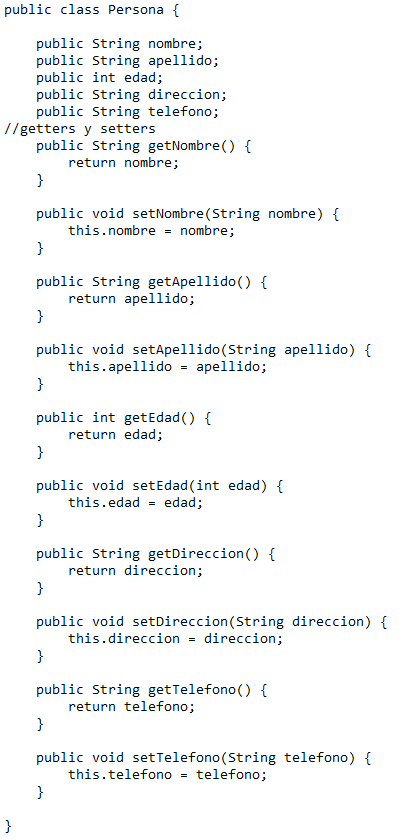
**Data clumps**

**Código original:**





**Código modificado:**



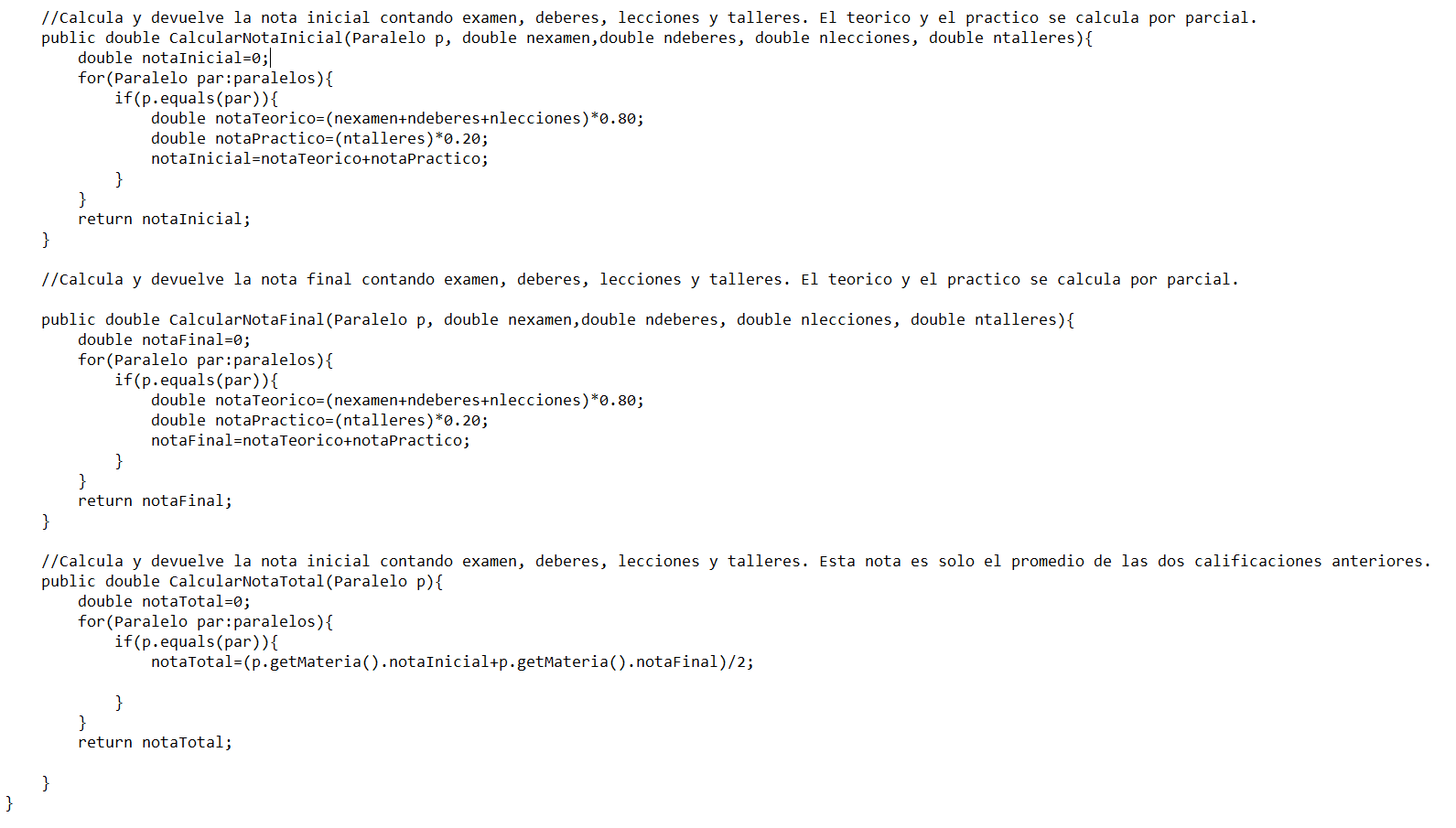
**Consecuencias:**

Mantener el código tal y como esta haría que este sea menos organizado pues las operaciones sobre datos particulares estarán distribuidas por toda la clase, además mantendría la clase extensa y difícil de comprender.

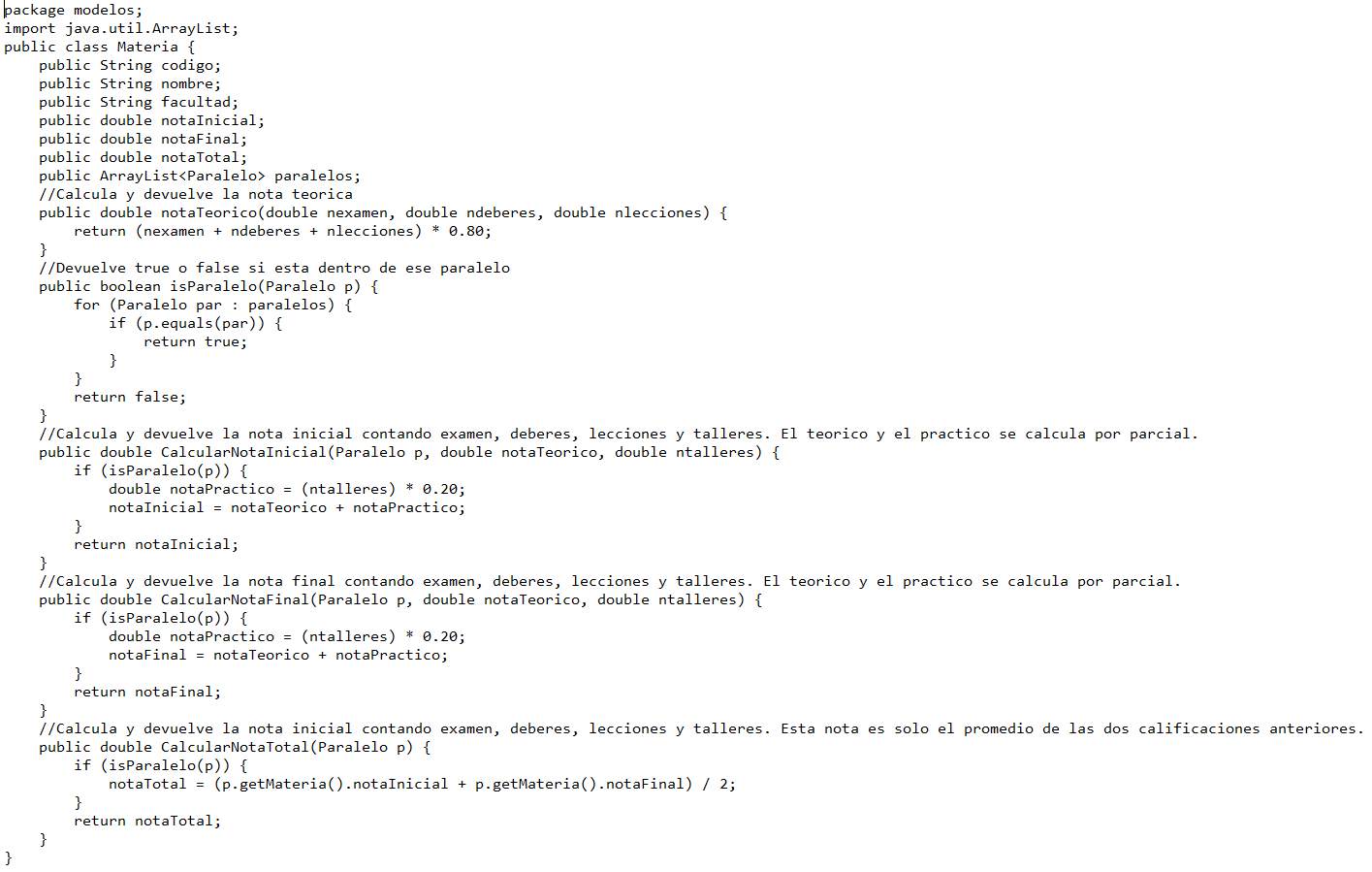
**Técnicas de refactorización:** Extract SuperClass

**Temporary Field**

**Código original:**



**Código modificado:**



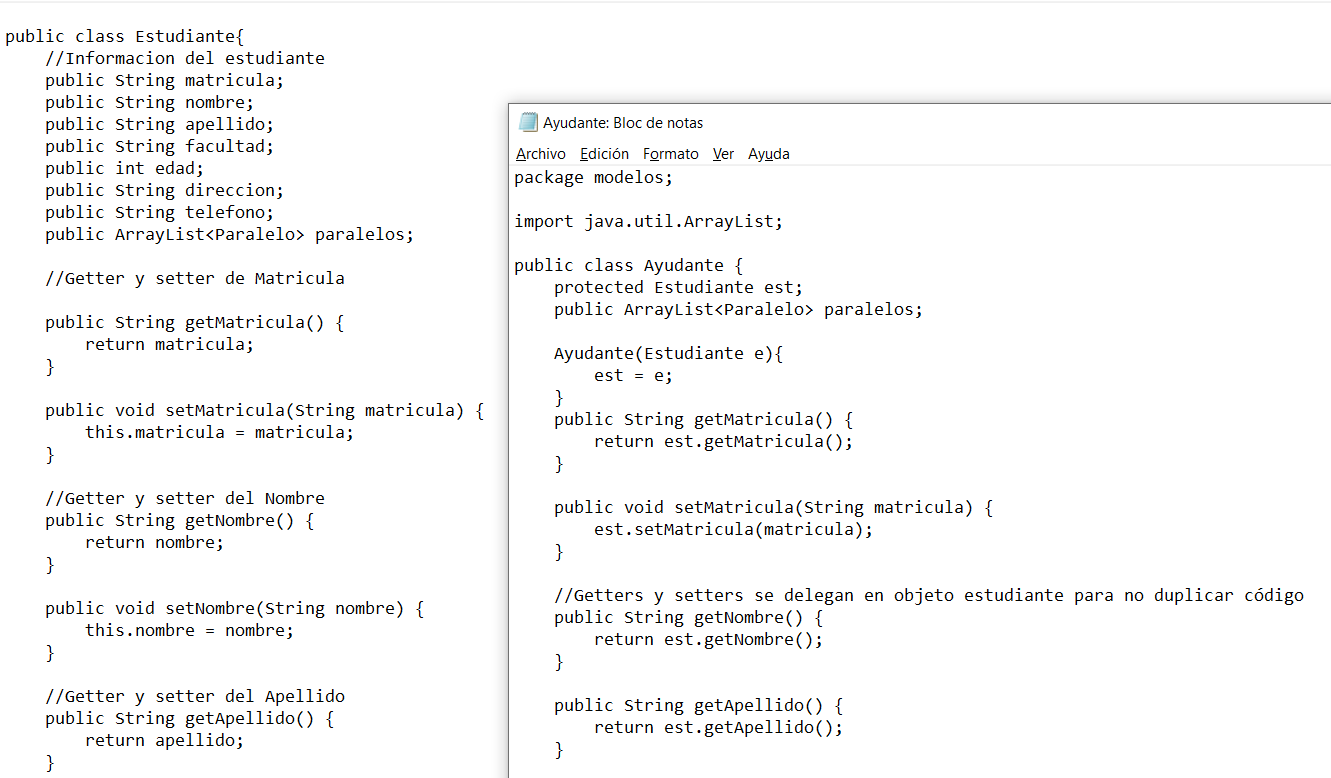
**Consecuencias:**

Mantener el código así, existiría un pequeño desperdicio de memoria al crear variables que no son necesarias, además añadiría líneas innecesarias al código, haciéndolo más extensos.

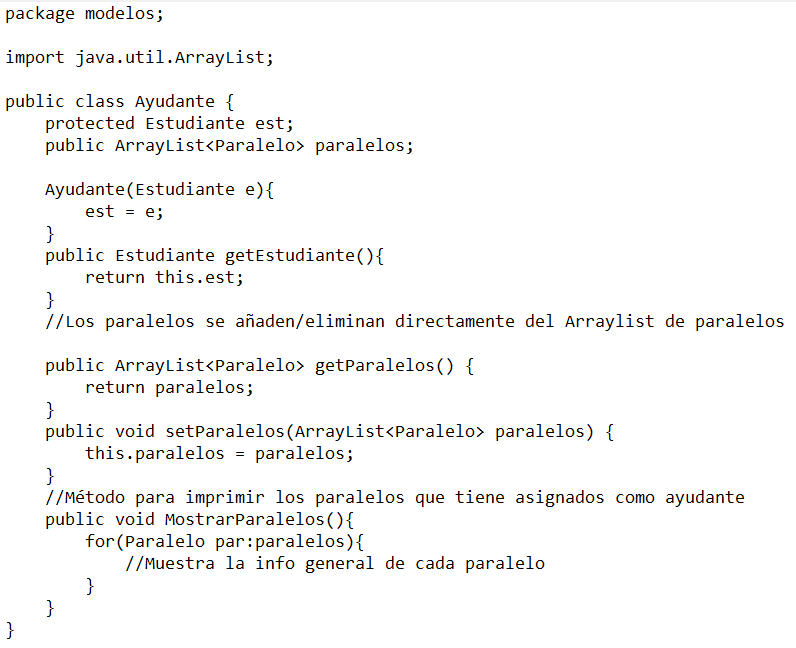
**Técnicas de refactorización:** Introduce Null Object

**Alternative Classes with Different Interfaces**

**Código original:**



**Código modificado:**



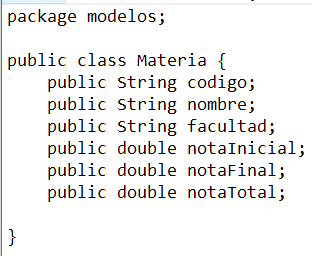
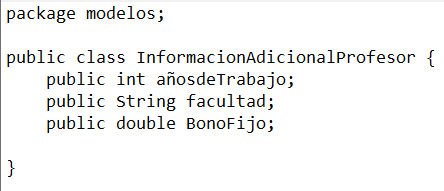
**Consecuencias:**

De mantener el código así, las clases serían más extensas y tendrían varios métodos que hacen prácticamente lo mismo.

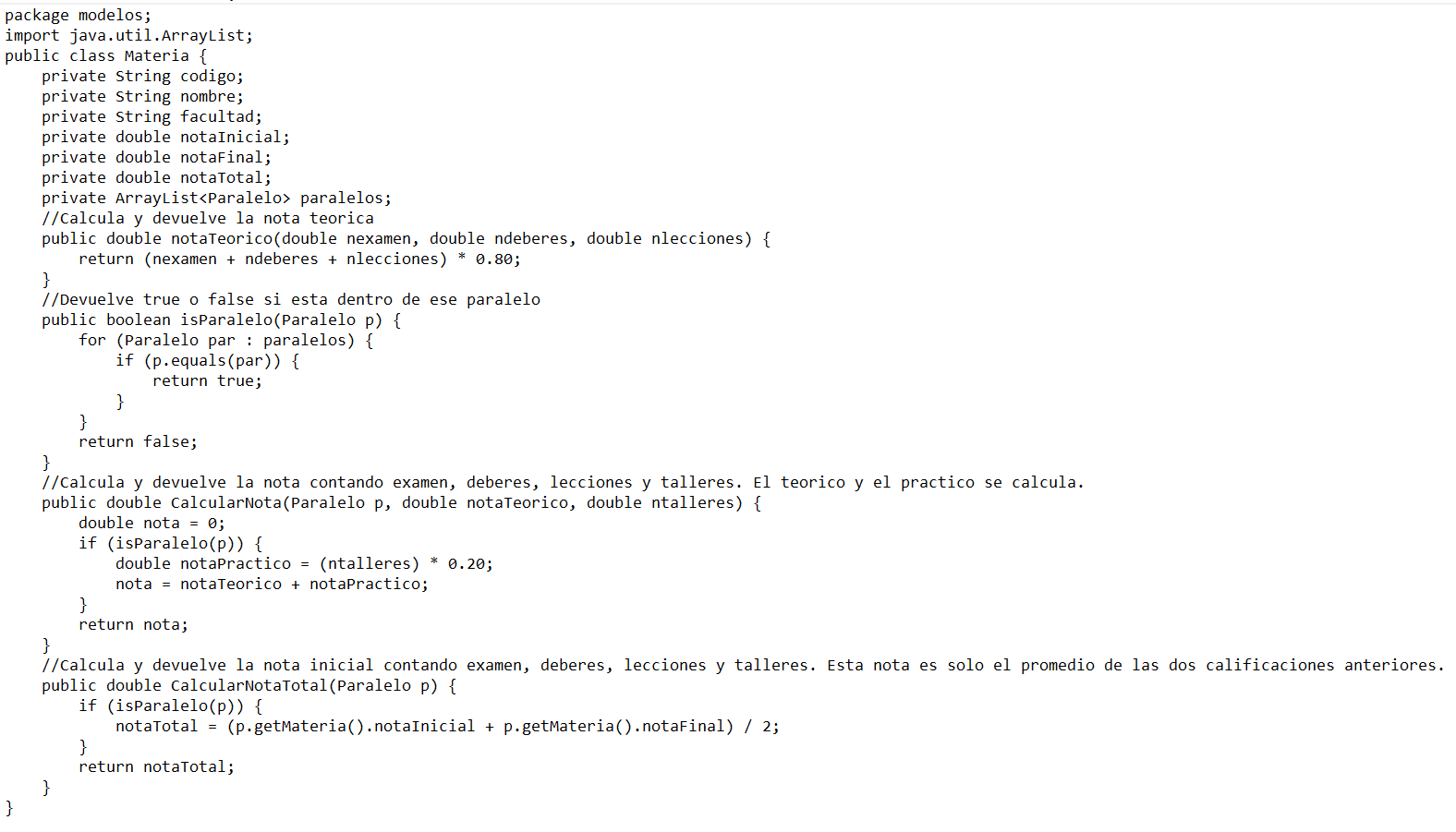
**Técnicas de refactorización:** Move Method, Add Parameter and Parameterize Method

**Data class**

**Código original:**



**Código modificado:**



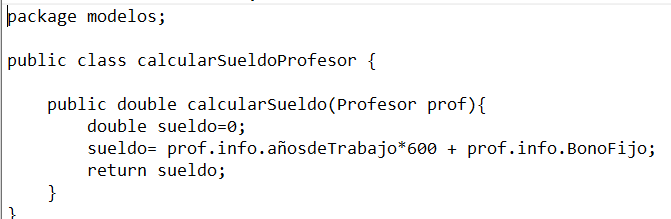
**Consecuencias:**

Mantener el código así haría más probable la duplicación de código, además que tendría datos particulares por todo el código en vez de tenerlos en un solo lugar, por lo que haría al código más difícil de comprender.

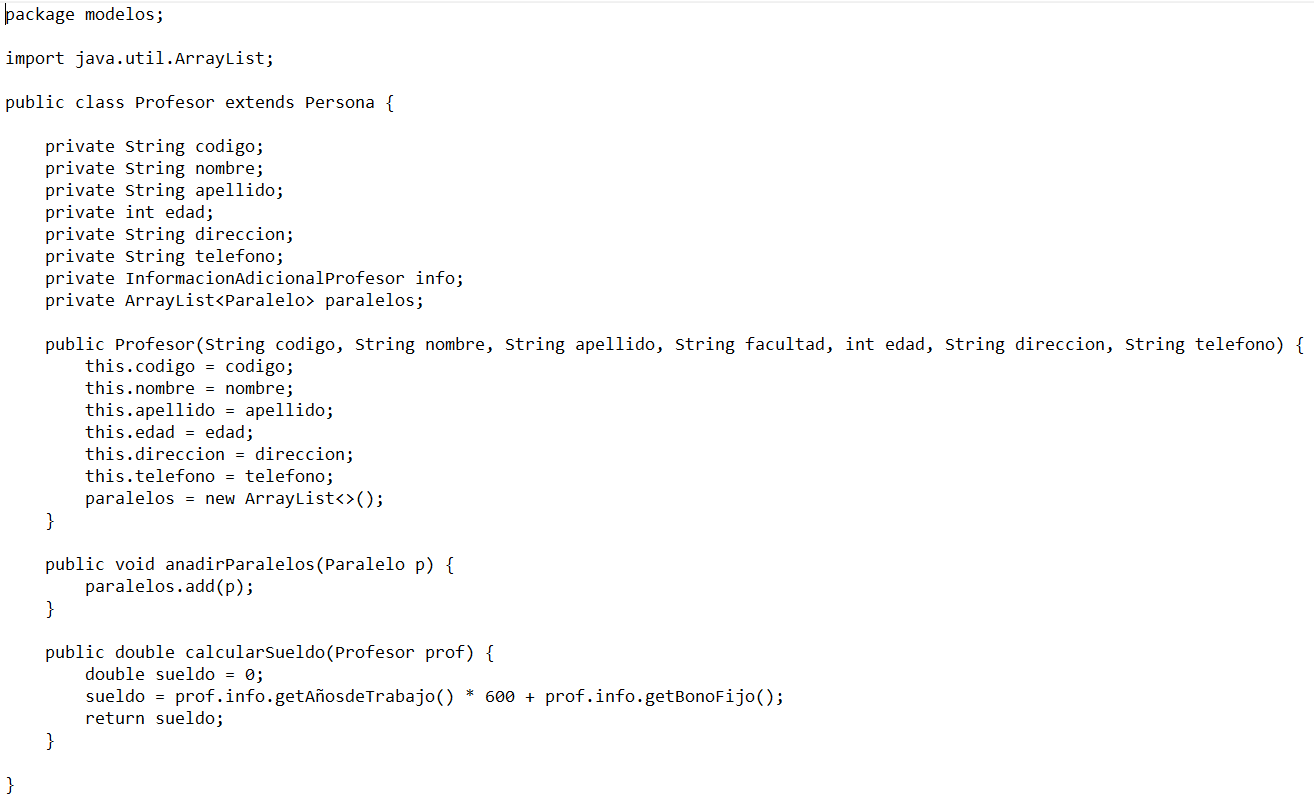
**Técnicas de refactorización:** Move method

**Feature envy**

**Código original:**



**Código modificado:**



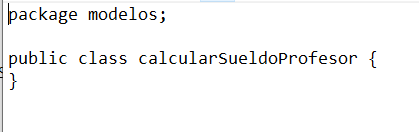
**Consecuencias:**

Mantener el código así haría que este esté mal organizado, además que aumenta la probabilidad de duplicación de código.

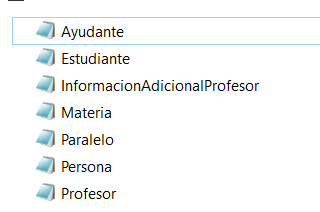
**Técnicas de refactorización:** Move Method

**Lazy class**

**Código original:**



**Código modificado:**



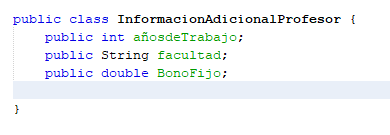
**Consecuencias:**

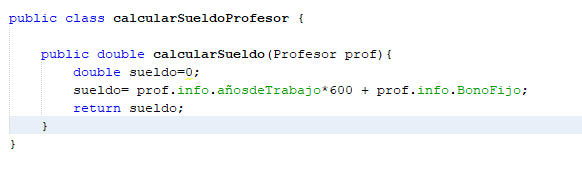
Mantener clases consume recursos y tiempo, por lo que mantenerla complicaría el mantenimiento, además extendería el código.

**Técnicas de refactorización:** Inline Class

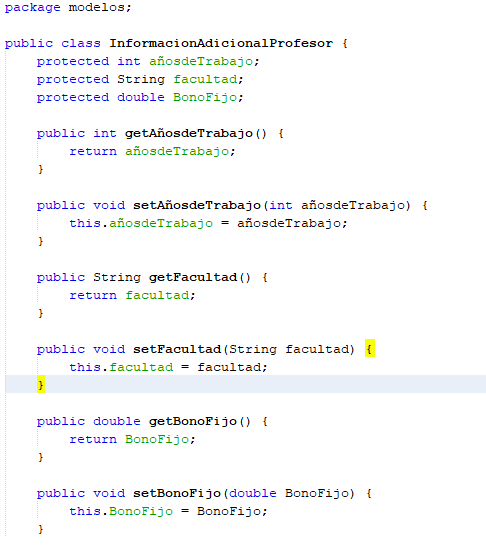
**Inappropiate Intimacy**

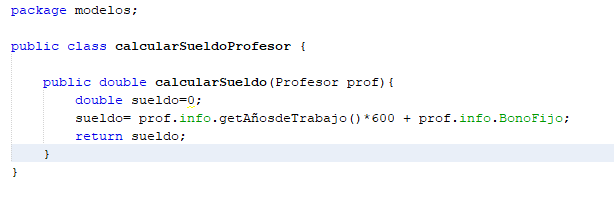
**Código original:**





**Código modificado:**





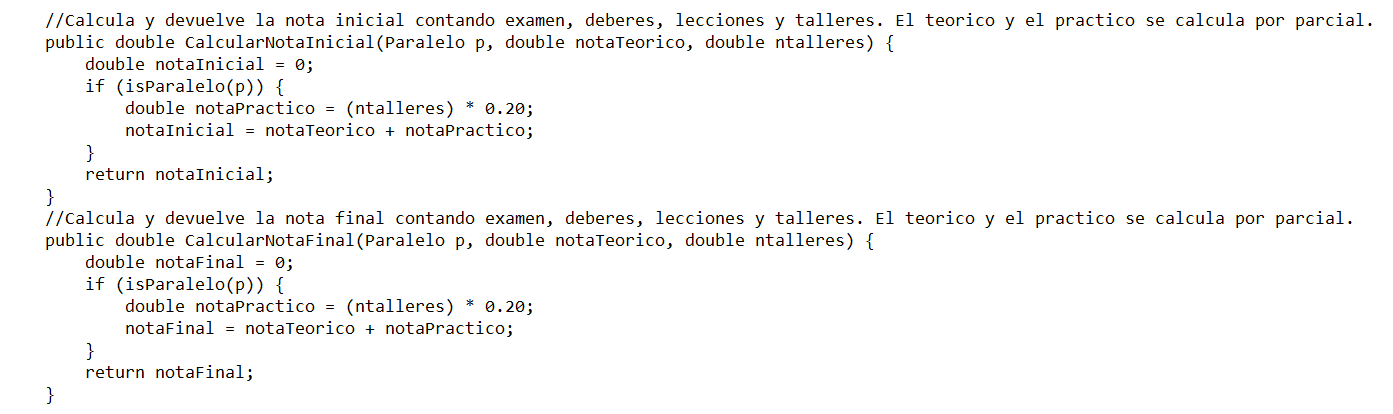
**Consecuencias:**

Mantener el código así complicaría el soporte y la reutilización de código, además se tendría un código más desorganizado y menos seguro.

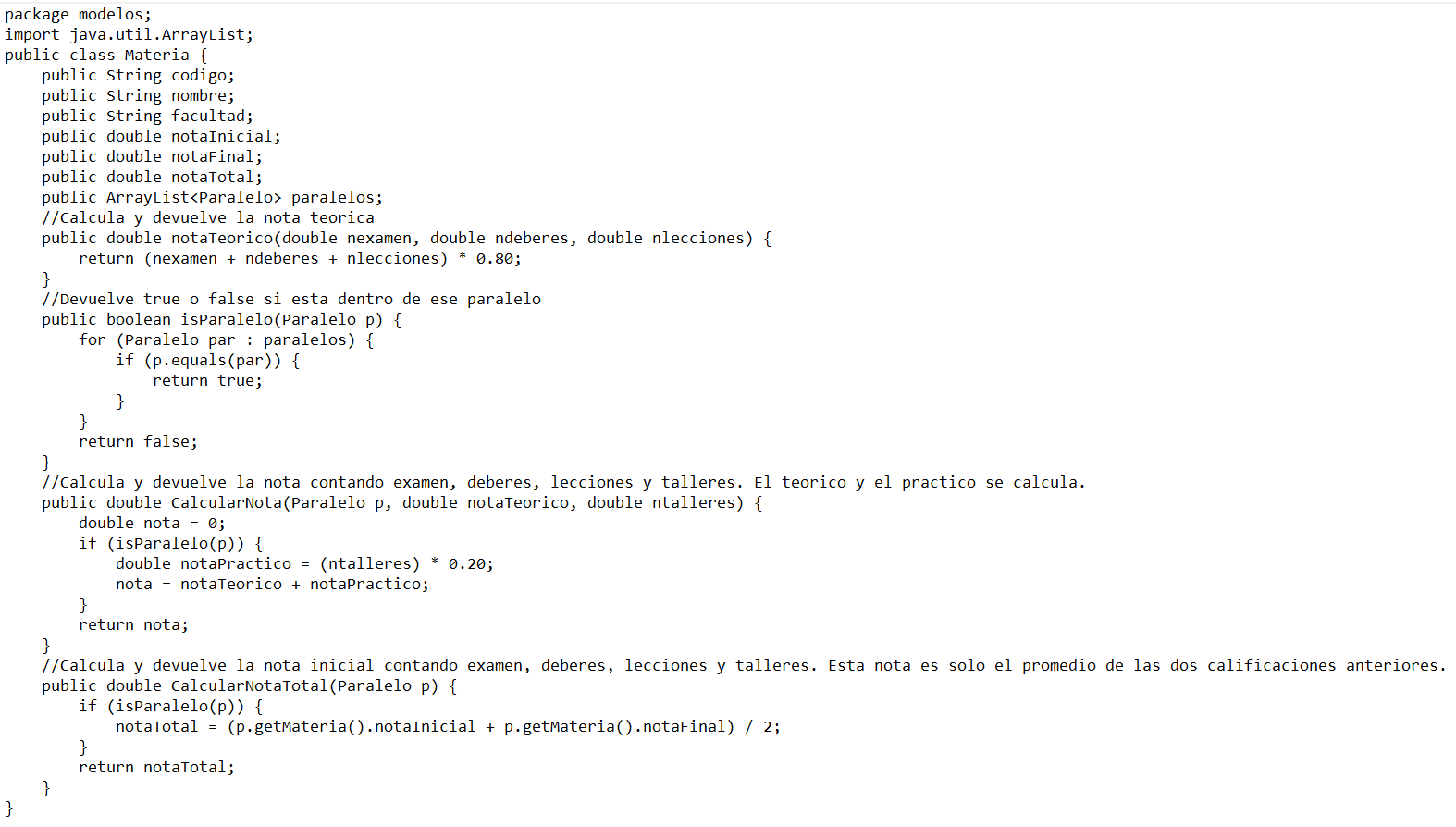
**Técnicas de refactorización:** Hide Delegate

**Duplicated Code**

**Código original:**



**Código modificado:**



**Consecuencias:**

Mantener el código así haría que exista código duplicado por lo cual la clase crecería sin necesidad, se volvería mas complejo y as costoso de mantener.

**Técnicas de refactorización:** Extract Method