**ESCUELA SUPERIOR POLITECICA DEL LITORAL**

**Taller Refactoring**

**Grupo 1**

**Integrantes:**

**- Alcivar Castro José Emanuel**

**Link Repositorio:**

**- https://github.com/Jose-Alcivar-C/TallerRefactoringG1**

**Data class**

La clase “InformacionAdicionalProfesor” solamente contiene atributos que perfectamente pueden pertenecer directamente a la clase “Profesor”. En esta clase no hay métodos que contengan acciones adicionales, solamente contiene campos que pertenecen a la clase “Profesor”, podemos eliminar la clase “InformacionAdicionalProfesor” y mover esos atributos a la clase “Profesor”. Además, podemos ver que en ambas clases estos atributos están públicos (no están encapsulados), lo cual no debería ser así.

**Consecuencias**

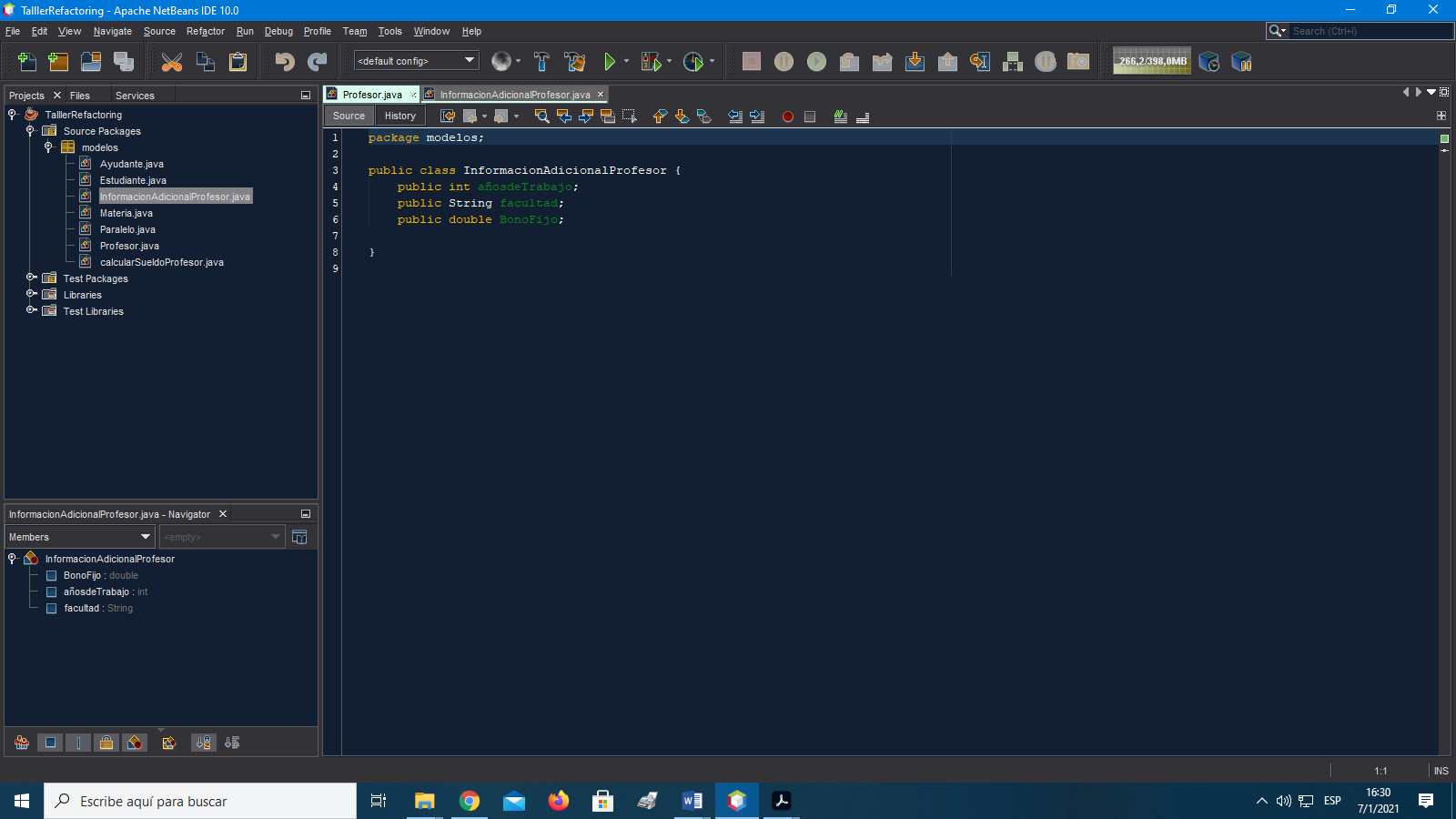
Sería necesario crear más métodos adicionales puesto que los atributos que se usarán se encuentran repartidos entre dos clases distintas, incluso corremos el riesgo de duplicar código en el programa.

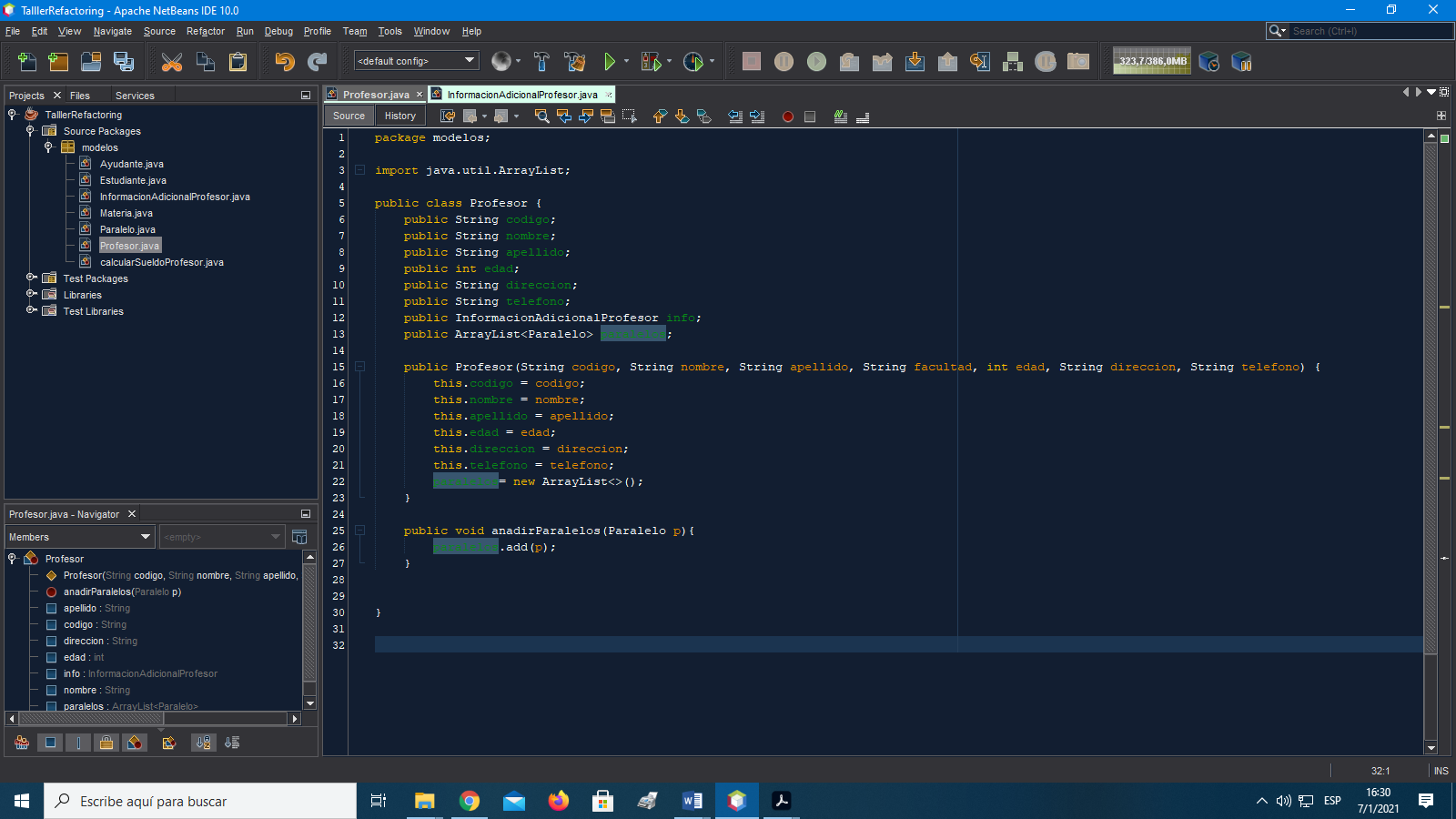
**Técnicas usadas**

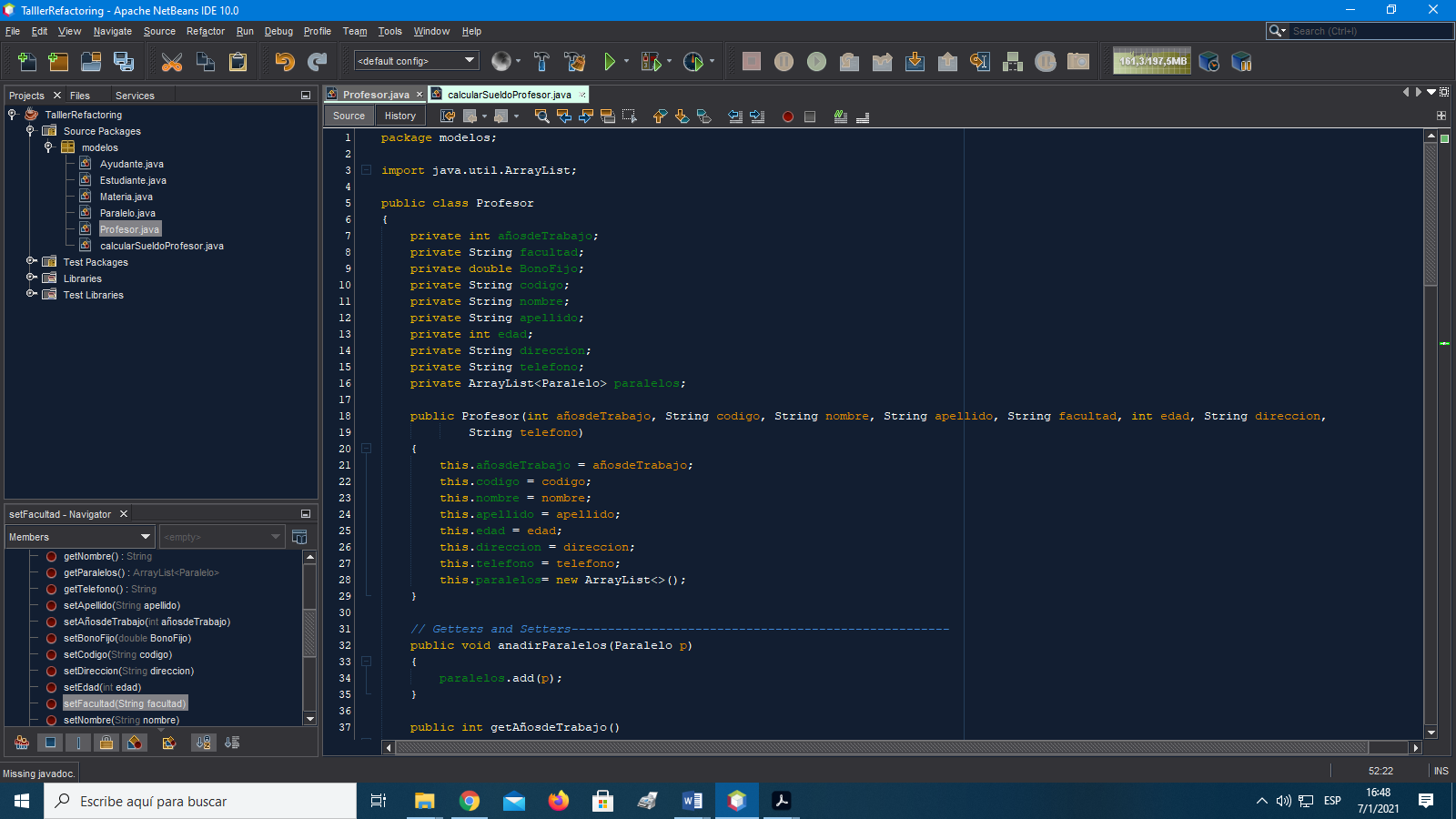
- Move Fields

- Encapsulate Fields

**Código inicial**

****

****

**Código refactorizado**

**Innapropiate Intimacy**

La clase “calcularSueldoProfesor” contiene un método que accede directamente a los atributos de otra clase, además de eso, podemos notar que este método no tendría que estar en esta clase, debido a que perfectamente podría estar en la clase “Profesor” y tomar de ahí los atributos, esto nos permite eliminar la clase “calcularSueldoProfesor”, puesto que está de más.

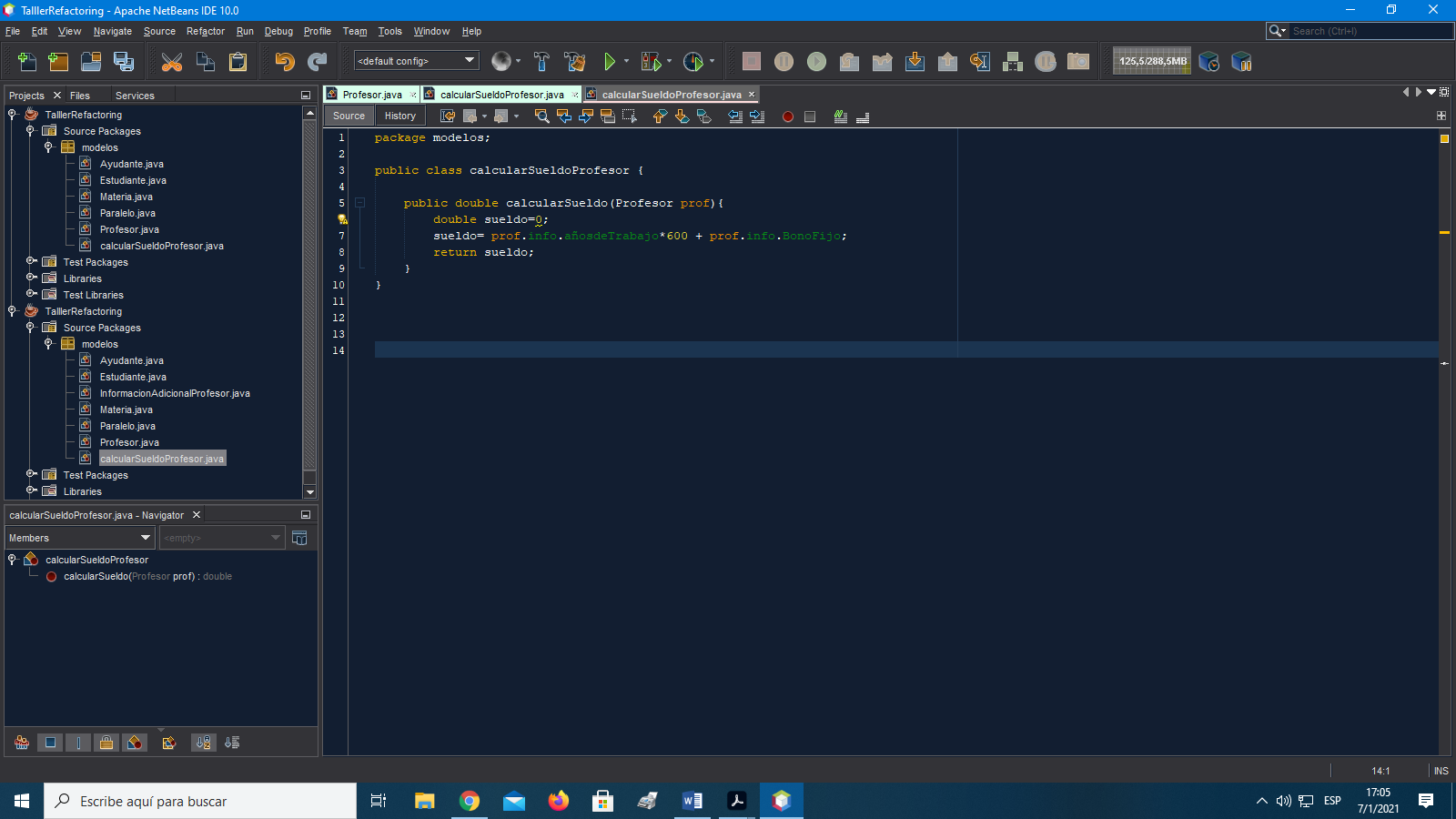
**Consecuencias**

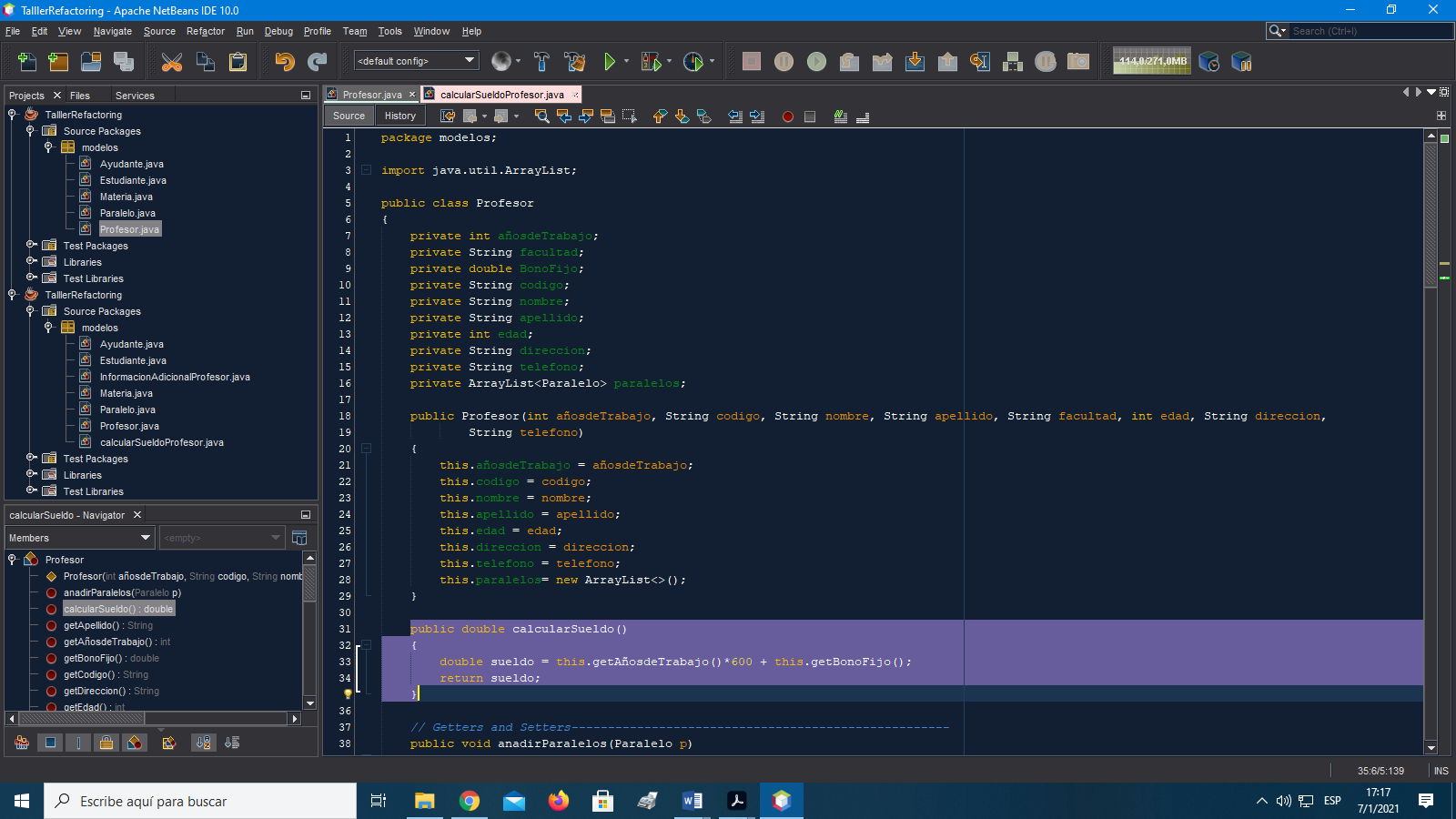
El código se vuelve más complejo de mantener, incluso se complica el control sobre los datos de cada clase debido a que se accede directamente a ellos.

**Técnicas usadas**

- Encapsulate Fields

- Move Method

**Código inicial**

**Código Refactorizado**