**Taller Refactoring**

**Diseño de Software**

**Integrantes:**

* Hatus Pellegrini
* Joby Farra
* Josue Montoya
* Oscar Lucas

**ESPOL 2T-2020**



Tabla de contenido

[1. Code Smells: Duplicate Code 3](#_Toc60952315)

[1.1. Consecuencias 3](#_Toc60952316)

[1.2. Técnica de Refactoring 3](#_Toc60952317)

[1.3. Código Inicial 3](#_Toc60952318)

[1.4. Código Final 3](#_Toc60952319)

[2. Code Smells: Feacture Envy 4](#_Toc60952320)

[2.1. Consecuencia 4](#_Toc60952321)

[2.2. Técnica de Refactorización 4](#_Toc60952322)

[2.3. Código Inicial 4](#_Toc60952323)

[2.4. Código Final 5](#_Toc60952324)

[3. Code Smell: Comments 6](#_Toc60952325)

[3.1 Consecuencia 6](#_Toc60952326)

[3.2 Técnica de Refactorización 6](#_Toc60952327)

[3.3 Código Inicial 6](#_Toc60952328)

[3.4 Código Final 6](#_Toc60952329)

[4 Code Smell: Long Class 7](#_Toc60952330)

[4.1 Consecuencias 7](#_Toc60952331)

[4.2 Técnica de Refactorización 7](#_Toc60952332)

[4.3 Código Inicial 7](#_Toc60952333)

[4.4 Código Final 7](#_Toc60952334)

[5. Code Smell: Data class 8](#_Toc60952335)

[5.1 Código initial / code smell 8](#_Toc60952336)

[5.2 Consecuencias 8](#_Toc60952337)

[5.3 Técnicas de refactorización 8](#_Toc60952338)

[5.4 Código modificado 9](#_Toc60952339)

[6. Code Smell: Inappropiate intimacy 10](#_Toc60952340)

[6.1 Consecuencia 10](#_Toc60952341)

[6.2 Técnica de Refactorización 10](#_Toc60952342)

[6.3 Código Inicial 10](#_Toc60952343)

[6.4 Código Modificado 10](#_Toc60952344)

# 1. Code Smells: Duplicate Code

## 1.1. Consecuencias

El code smells encontrado en la siguiente parte de código es duplicade code ya que existe un código fuente que ocurre mas de una vez, dentro de un programa o a través de diferentes programas. La consecuencia de esto radica en el espacio de memoria, el código duplicado ocupa memoria innecesaria lo cual se realizan procesos que hacen el código mas pesado y hace al programador confundirse.

## 1.2. Técnica de Refactoring

*Extract Method* es una técnica de Refactoring que nos ayuda cuando mas líneas se encuentren en un método, mas difícil averiguar qué hace el método. Entonces agregamos en una función todo ese código que cuando se necesita se llame.

## 1.3. Código Inicial

Graphical user interface, text

Description automatically generated

## 1.4. Código Final

Graphical user interface, text, application

Description automatically generated

# 2. Code Smells: Feacture Envy

## 2.1. Consecuencia

El code smells encontrado en el siguiente bloque de código es Feature Envy el cual lo identificamos cuando un objeto está mas interesado en otro que en su mismo. Existe mucho acoplamiento entre la clase ayudante y estudiante, la consecuencia con esto es que si se elimina una parte de código en la clase Estudiante puede afectar al funcionamiento de la clase Ayudante.

## 2.2. Técnica de Refactorización

*Reemplazar Delegación por Herencia* es una técnica de refactoring dice que podemos usar la herencia para para lograr que la clase estudiante y la clase ayudante trabajen en códigos separados para que no haya necesidad de que haya código duplicado. Además, se logra que la clase ayudante se construya con menos código y sea posible mantenerlo a lo largo del tiempo.

## 2.3. Código Inicial

Graphical user interface, text, application, email

Description automatically generated

## 2.4. Código Final

Graphical user interface, text, application, email

Description automatically generated

# 3. Code Smell: Comments

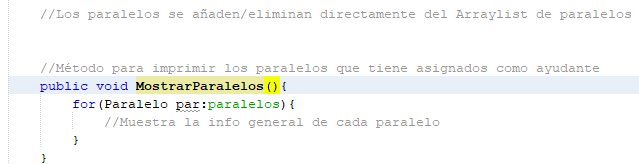
## 3.1 Consecuencia

El code smell se evidencia en la clase Ayudante, donde para el metodo MostrarParalelos () se explica con 3 comments su funcionamiento. Esto no debería ser así, dado que el código debe desarrollarse de manera que pueda ser entendido por cualquiera que lo vea sin necesidad de comentarios adicionales.

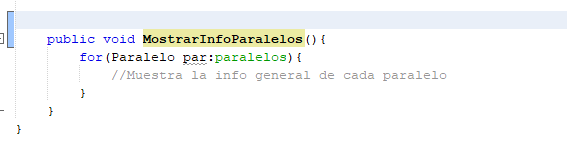
## 3.2 Técnica de Refactorización

Para solucionarlo, se utiliza la técnica de Rename Method, en donde se cambia el nombre del metodo por uno más intuitivo sobre su funcionamiento, para que los comentarios no sean necesarios para su compresión.

## 3.3 Código Inicial



## 3.4 Código Final



# 4 Code Smell: Long Class

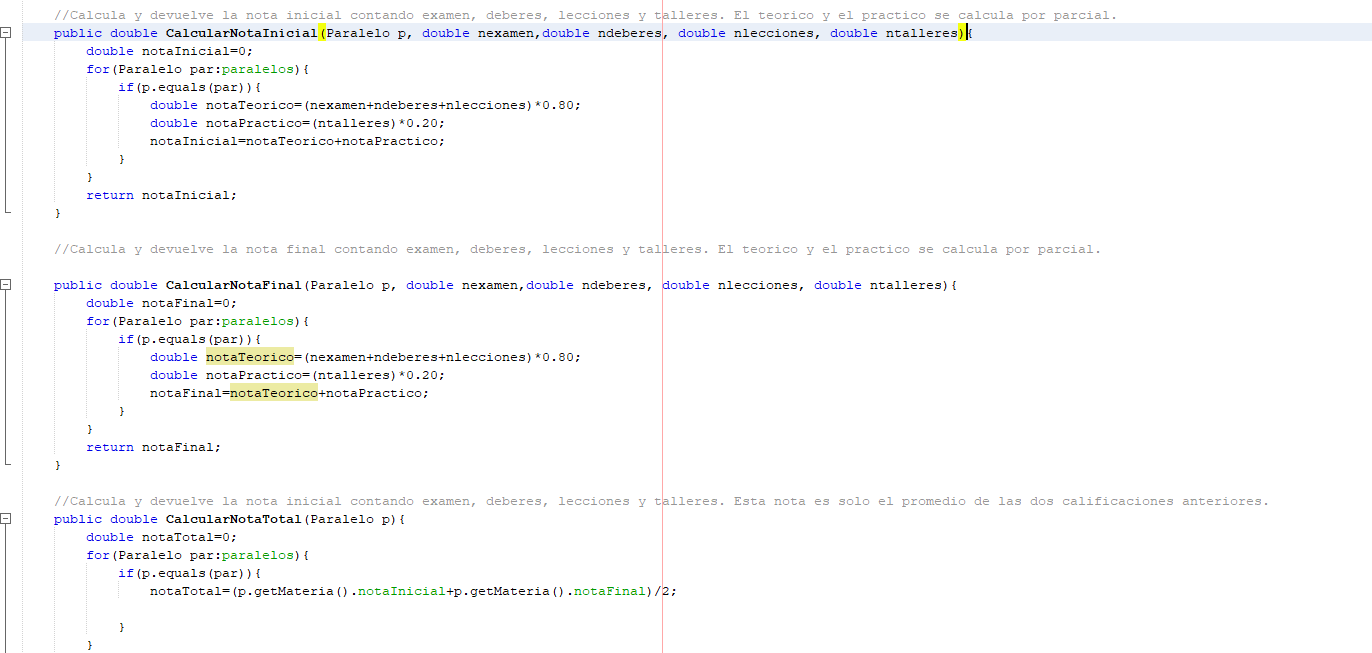
## 4.1 Consecuencias

Este code smell se puede identificar en la clase Estudiante, la cual es sumamente extensa y puede llegar a causar confusión al momento de leer el código o de tratar de solucionar algún bug.

## 4.2 Técnica de Refactorización

Para solucionar este code Smell se utiliza la técnica de refactorización de Extract Class, en donde parte del código dentro de la clase Estudiante se separa a otro componente, en este caso una clase Calculadora Denotas, la cual se encarga únicamente de calcular la nota dado un estudiante.

## 4.3 Código Inicial

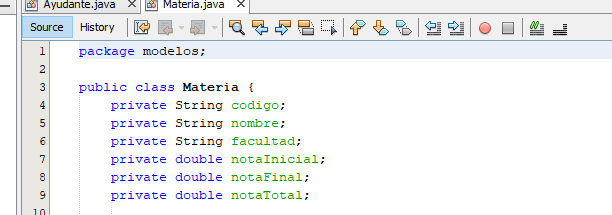


## 4.4 Código Final



# 5. Code Smell: Data class

## 5.1 Código initial / code smell



## 5.2 Consecuencias

En la clase Materia se puede evidenciar 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. 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.

## 5.3 Técnicas de refactorización

Como solución para 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.

## 5.4 Código modificado

# 6. Code Smell: Inappropiate intimacy

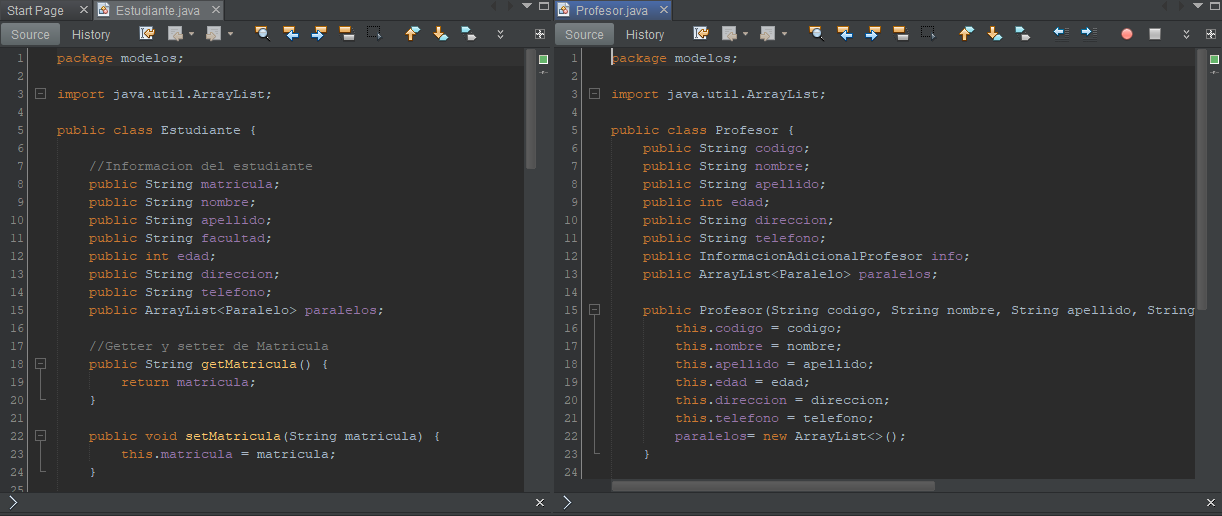
## 6.1 Consecuencia

En la clase Estudiante y Profesor, se puede apreciar que los atributos que poseen estas clases tienen un modificador de acceso “public”, por lo que permite que estos atributos sean accedidos desde clases externas. Por lo que este code smell se lo conoce como Inappropiate intimacy.

## 6.2 Técnica de Refactorización

Para solucionar este code smell encontrado en estas clases, se puede aplicar la técnica de refactorización llamada “Encapsulated Field”, la cual nos permite cambiar los modificadores de accesos de los atributos a “private” y así permitir el acceso solo a clases derivadas, además permitir el uso por medio de los get and setters.

## 6.3 Código Inicial



## 6.4 Código Modificado

