Taller08-Refactoring

Integrantes:

Irving Macías

Roberto Encalada

Christopher Acosta



Tabla de contenido

[Duplicate Code 3](#_Toc122657746)

[Inappropriate Intimacy 4](#_Toc122657747)

[Data Clumps 5](#_Toc122657748)

[Data Class 6](#_Toc122657749)

[Speculative Generality 7](#_Toc122657750)

[Dead Code 8](#_Toc122657751)

[Feature Envy 9](#_Toc122657752)

# Duplicate Code

Código Antes:

Consecuencias:

* El código duplicado puede hacer que el código sea algo redundante para leer.
* Hace que la clase sea innecesariamente larga.

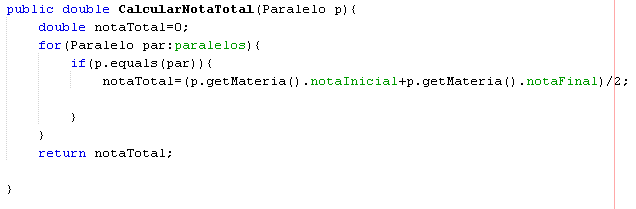
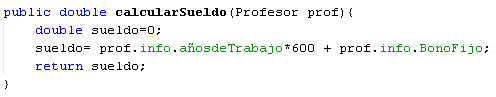
Técnica de Refactorización por utilizar: Extract Method

Código Después:

Imagen de la pantalla de un celular con texto

Descripción generada automáticamente con confianza media

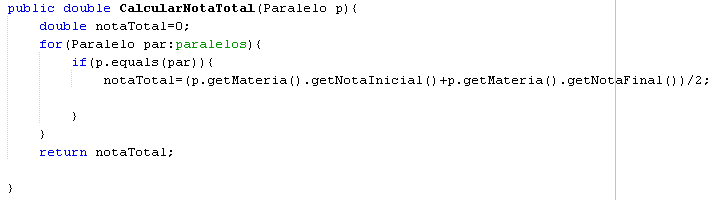
# Inappropriate Intimacy

Código Antes:

Consecuencias:

* Los atributos públicos pueden ser accedidos por otras personas y modificar su información

Técnica de Refactorización por utilizar: Move Field, Move Method y Inline Method

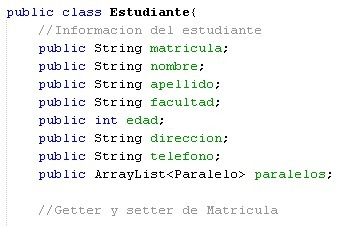
Código Después:

Texto

Descripción generada automáticamente con confianza mediaTexto

Descripción generada automáticamente

# Data Clumps

Código Antes

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Consecuencia:

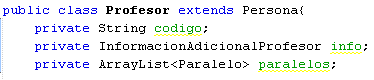
* A medida que el programa se expande, repetiremos código de manera innecesaria

Técnica de refactorización por usar: Extract Class

Código Después:

Imagen que contiene Texto

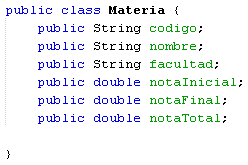
Descripción generada automáticamente



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# Data Class

Código antes:

Consecuencias:

* Dado que es una clase que sólo guarda atributos públicos, entonces se puede caer en la tentación de duplicar código con tal de hacer una operación específica sobre esos atributos. De esta forma, se tendría código disperso e idéntico a lo largo del programa.

Técnica de refactorización por usar: Encapsulate Field

Código después:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# Speculative Generality

Código Antes:

Interfaz de usuario gráfica, Texto

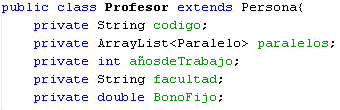
Descripción generada automáticamente

Consecuencias:

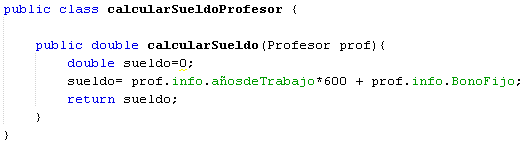
* Se hace implementación demás que puede llegar a confundir a una persona que trate de leer el código
* Se pueden llegar a crear clases vagas

Técnica de refactorización por usar: Move Field

Código después:



# Dead Code

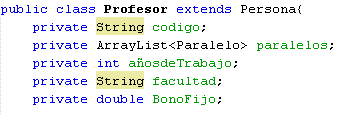
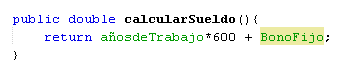
Código antes: 

Consecuencias:

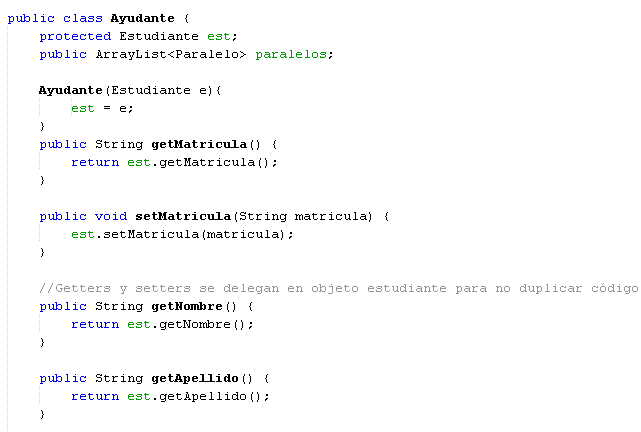
* La clase calcularSueldoProfesor sólo está pensada para calcular sueldo de un profesor, por lo que se podría mover dicho método dentro de profesor o simplemente eliminar esta clase si este método ya no se va a utilizar.

Técnica de refactorización por usar: [Inline Class](https://refactoring.guru/es/inline-class) y Inline Method

Código después:



# Feature Envy

Código antes:

Consecuencias:

* Se está creando código demás que solo ocupa más espacio de memoria

Técnica de refactorización por usar: Replace delegation with Inheritance

