**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**DISEÑO DE SOFTWARE**

**TALLER 8 Refactoring**

**PARALELO 102**

**INTERGRANTES:**

DAYANA CUMBA

ROBERT SÁNCHEZ

ALEXANDER PAUTA



Tabla de contenido

[**1.** **Code smell: Inapropiate Intimacy** 3](#_Toc92391749)

[**Consecuencias:** 3](#_Toc92391750)

[**Técnicas de refactorización:** 4](#_Toc92391751)

[**2.** **Code Smell: Long Parameter List** 4](#_Toc92391752)

[**Consecuencias:** 4](#_Toc92391753)

[**Técnicas de refactorización:** 5](#_Toc92391754)

[**3.** **Code Smells: Data Clumps** 5](#_Toc92391755)

[**Consecuencias:** 5](#_Toc92391756)

[**Técnicas de refactorización:** 6](#_Toc92391757)

[**4.** **Code Smells: Tempory Field** 6](#_Toc92391758)

[**Consecuencias:** 6](#_Toc92391759)

[**Técnicas de refactorización:** 6](#_Toc92391760)

[**5.** **Code Smells: Comments** 6](#_Toc92391761)

[**Consecuencias:** 6](#_Toc92391762)

[**Técnicas de refactorización:** 7](#_Toc92391763)

[**6.** **Code Smells: Lazy Class** 7](#_Toc92391764)

[**Consecuencias:** 7](#_Toc92391765)

[**Técnicas de refactorización:** 8](#_Toc92391766)

# **Code smell: Inapropiate Intimacy**

En este caso la clase ayudante necesita acceder a los parámetros de la clase Estudiante para su funcionamiento generando un acoplamiento entre clases.

## **Consecuencias:**

Esto puede generar que el desarrollador no pueda entender de una manera correcta ninguna de las dos clases. En caso de que se desee realizar un cambio en la clase Estudiante afectaría directamente a la clase Ayudante.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**Captura inicial:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## **Técnicas de refactorización:**

En este caso lo más optimo seria extraer la clase y hacer uso de la herencia, se toma en cuenta al ayudante como un caso “especial” de estudiante haciendo que extienda directamente de dicha clase.

**Captura final:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

# **Code Smell: Long Parameter List**

## **Consecuencias:**

Los métodos para calcular notas tienen demasiados parámetros lo que hace que el método no sea entendible.

**Captura inicial:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## **Técnicas de refactorización:**

Lo ideal es crear una clase que se encargue de maneja cada parámetro del método calcular notas, para que sea más sencillo acceder a ellas y evitar la sobrecarga en el método.

**Captura final:**

**Nota.java**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Estudiante.java**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

# **Code Smells: Data Clumps**

## **Consecuencias:**

Se tendrían variables repetidas como el nombre y apellidos entre las clases de estudiante y profesor incluido los métodos getters y setters por lo que tendríamos un código extenso.

* **Captura inicial:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## **Técnicas de refactorización:**

Para mejorarlo creamos una clase Persona que contiene las variables y métodos que se repetían en ambas clases.

* **Captura final:**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

# **Code Smells: Tempory Field**

## **Consecuencias:**

Los campos temporales obtienen sus valores (y, por lo tanto, los objetos los necesitan) solo en determinadas circunstancias. Fuera de estas circunstancias, están vacías. Variables que no tienen sentido crearlas.

**Captura inicial:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## **Técnicas de refactorización:**

Podemos eliminar las variables innecesarias y poner el código en una sola sentencia.

**Captura final:**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

# **Code Smells: Comments**

## **Consecuencias:**

El código está lleno de comentarios innecesarios en métodos muy obvios cuando solo se deben usar en métodos complejos.

Texto

Descripción generada automáticamente**Captura inicial:**

## **Técnicas de refactorización:**

Se eliminan los comentarios innecesarios de los getters y setters solo se conservan en los algunos métodos medianamente complejos.

**Captura final:**

**Texto

Descripción generada automáticamente**

# **Code Smells: Lazy Class**

## **Consecuencias:**

Debido al pequeño aporte que genera esta clase, nos conviene remover y colocar ese método en otra clase, ya que nos ahorraríamos el mantenimiento que deba tener una clase nueva.

**Captura inicial:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## **Técnicas de refactorización:**

Ya que esta clase no presenta muchas funcionalidades, e incluso la clase utiliza los datos del Profesor que recibe, podríamos mover el método a la clase Profesor y remover la clase.

**Captura final:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente