# 

**11-01-2020**

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**DISEÑO DE SOFTWARE**

**REFACTORING**

**AUTORA:**

Moyano Salazar Victor

**DOCENTE**:

Jurado David

**PARALELO:** 1

**Guayaquil-Ecuador**

**2019-2S**

# Feature Envy / Característica Envidiada

Este code smell se da cuando una clase trata de acceder los campos de otra clase para realizar una ejecución y no es la clase dueña de los campos quien realiza dicha ejecución.

**Técnicas de refactorización**

* Movimiento. – Esta técnica mueve un método de una clase a otra, donde sea adecuada su existencia.
* Extracción. – Esta técnica se usa cuando una instancia está usando atributos de otro objeto.

# Primitive Obsession / Obsesión Primitiva

Se da cuando una clase hace demasiado uso a datos de tipo primitivo, es ahí donde se debe observar si estos datos pueden llegar a formar un dato de tipo Object (clase). Aquí abunda la falta de abstracción.

**Técnicas de refactorización**

* Extraer Clase: Se crea una nueva clase con los datos de tipo primitivo como atributos.
* Remplazar parámetros: Los métodos donde se llamaban a los datos primitivos ahora llamaran a la instancia necesaria.

# Refused Bequest / Legado Rechazado

Se da cuando una clase hija no hace uso de todo los comportamientos o atributos heredados de la clase padre. Además, también se viola con el principio de sustitución de Liskov, dado a que una instancia de la clase hija no podría sustituir a una instancia de la clase padre.

**Técnicas de refactorización**

* Extracción de Superclase: Se extrae la funcionalidad que es compartida y se crea una superclase abstracta común para las clases involucradas.

# Switch Statements/ Sentencias Switch

Se da cuando existe un uso excesivo de sentencias switch y de ifs anidados. El problema se da cuando se debe agregar otra condición, ya que siempre será necesario modificar la sentencia switch.

**Técnicas de refactorización**

* Extraer método y Mover método: Se aísla el switch y se reubica en la clase donde realimente es necesario.
* Reemplazar condicional con polimorfismo: Se realiza la implementación necesaria para cada tipo de objeto referenciado en los case.

# Data Clump/ Grupo de datos

Se da cuando grupos de datos se mantienen juntos ya sea al estar como parámetros para un método. Al crear una clase que contengan estos datos, se ahorra código y se hace más simple el mismo.

**Técnicas de refactorización**

* Extraer Clase: Se crea una clase la cual tendrá los atributos y la funcionalidad que realmente es necesaria para la ejecución del código.
* Introducir Objeto como parámetro: En métodos donde se usen grupos de datos como parámetros o se tenga un gran número de parámetros, es mejor remplazar estos parámetros creando un objeto que contenga estos datos.

# Bibliografía

Guru. (s.f.). *refactoring*. Obtenido de https://refactoring.guru/refactoring/smells

*wiki.c2.com*. (12 de 12 de 2014). Obtenido de http://wiki.c2.com/?CodeSmell

Enlace a GitHub:

https://github.com/vjmoyano/trabajoautonomo-refactoringMoyano.git