# 

**01-11-2020**

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**DISEÑO DE SOFTWARE**

**REFACTORING**

ENERO DEL 2020

**AUTORA:**

Holguín Joselyn

(jeholgui@espol.edu.ec)

**DOCENTE**:

Jurado David

**PARALELO:** 1

**Guayaquil-Ecuador**

**2019-2S**

**CONTENIDO**

[**REFACTORING** 2](#_Toc29679022)

[**1.** **FEATURE ENVY – ENVIDIA CARACTERÍSTICA** 2](#_Toc29679023)

[**2.** **PRIMITIVE OBSESSION – OBSESIÓN PRIMITIVA** 3](#_Toc29679024)

[**3.** **SWITCH STATEMENTS – CAMBIAR DECLARACIONES** 3](#_Toc29679025)

[**4.** **DATA CLUMPS – GRUPOS DE DATOS** 4](#_Toc29679026)

[**5.** **SHOTGUN SURGERY – CIRUGÍA DE ESCOPETA** 5](#_Toc29679027)

[**6.** **ENLACE A GITHUB** 5](#_Toc29679028)

[**BIBLIOGRAFÍA** 6](#_Toc29679029)

# **REFACTORING**

# **FEATURE ENVY – ENVIDIA CARACTERÍSTICA**

**DESCRIPCIÓN:**

Es cuando un método tiene acceso a datos que son pertenecientes a otro objeto, inclusive más que a sus propios datos, esto sucede cuando los campos son cambiados a una clase de datos (Gurú, 2014).

**TÉCNICAS DE REFACTORIZACIÓN:**

* **Método de movimiento:** Se emplea cuando un método se debe cambiar a un lugar distinto del actual (Sourcemaking, 2007).
* **Método de extracción:** Se usa moviendo el código de la parte referente a un método que tiene acceso a datos de otro objeto.

# **PRIMITIVE OBSESSION – OBSESIÓN PRIMITIVA**

**DESCRIPCIÓN:**

Utilización de datos primitivos en vez de métodos y objetos, uso de constantes para la codificación de tipo (ADMIN\_USER=1, enum) y uso de cadenas para emplear como nombres de campo con el fin de usarlo en las matrices de datos (Gurú, 2014).

**TÉCNICAS DE REFACTORIZACIÓN:**

* **Introducir objeto de parámetro o Conservar objeto completo:** Esto se usa en el caso de que los valores de los campos primitivos se empleen en métodos, explícitamente en sus parámetros.
* **Reemplazar código de tipo con clase / Reemplazar código de tipo con subclases/ Reemplazar código de tipo con estado / estrategia:** Esto se debe usar en el caso de que los datos sean complicados y estén codificados en variables.
* **Reemplazar matriz con objeto:** Esto en el caso de que existan matrices entre las variables.

# **SWITCH STATEMENTS – CAMBIAR DECLARACIONES**

**DESCRIPCIÓN:**

Es cuando dentro de las clases se encuentran muchos switch y también se hallan if anidados o if-else (Gurú, 2014).

**TÉCNICAS DE REFACTORIZACIÓN:**

* **Extraer método y Mover método:** Esto sería en el caso de querer aislar el switch y colocarlo a la clase que realmente le correspondal (Sourcemaking, 2007).
* **Reemplazar código de tipo con subclases / Reemplazar código de tipo con estrategia/estado:** Se refiere a un código de tipo, es cuando se modifica el modo de execute time del programa.
* **Reemplazar condicional con polimorfismo:** Se usa luego de dar especificaciones referentes a la estructura de herencia.
* **Reemplazar parámetro con métodos explícitos:** Si existen muchas condiciones que llaman al mismo método, pero con parámetros distintos, en este punto se opta por dividir este método en métodos pequeños diferentes.
* **Introducir objeto nulo:** Si existen condicionales que contengan null.

# **DATA CLUMPS – GRUPOS DE DATOS**

**DESCRIPCIÓN:**

Es cuando existen diversas partes del código que contienen variables que pueden agruparse por ejemplo “Class conectBD” (Gurú, 2014).

**TÉCNICAS DE REFACTORIZACIÓN:**

* **Extract Class:** En este caso el comportamiento de la clase se divide (Sourcemaking, 2007).
* **Introduce Parameter Object:** Se emplean cuando los mismos grupos de datos son pasados en los parámetros de los métodos, estableciéndose como clase.
* **Preserve Whole Object:** Aquí se debe pasar el objeto de datos al método, en vez de sólo considerar campos de forma individual, esto sucede cuando los datos se trasladan a otros métodos.

# **SHOTGUN SURGERY – CIRUGÍA DE ESCOPETA**

**DESCRIPCIÓN:**

Es similar a cambio divergente, pero en sí es lo opuesto, en este caso es cuando se hace el mismo cambio, pero en diversas clases de manera simultánea (Gurú, 2014).

**TÉCNICAS DE REFACTORIZACIÓN:**

* **Move Method / Move Field:** Esto se hace con el fin de mover los comportamientos de la clase que ya existe a una sola clase y sino llegase a existir una clase apropiada para ellos, se debe crear una nueva clasel (Sourcemaking, 2007).
* **Inline Class:** Si al mover los métodos, la clase original queda vacía, trate de desaparecer esas clases redundantes.

# **ENLACE A GITHUB**

<https://github.com/elizabethHolguin/REFACTORING>

# **BIBLIOGRAFÍA**

*Gurú. (2014). Recuperado el 11 de enero de 2020, de https://refactoring.guru/refactoring/smells*

*Sourcemaking. (2007). Obtenido de https://sourcemaking.com/refactoring*