**Refactoring**

**ESPOL**

**FIEC**

**DISEÑO DE SOFTWARE**

**Juan Jose Loor**

**Enero 2019.**

# Nombre: Primitive Obsession - Obsesión Primitiva

## Descripción: Pertenece a los Bloaters y básicamente representa un problema en el código a largo plazo ya que comenzamos a emplear muchos datos primitivos con el fin de evitar crear más clases, pero el inconveniente esta cuando lo hacemos muchas veces y a largo plazo la clase se vuelve muy extensa y compleja.

## Técnicas :

* + Si los valores de los campos primitivos están siendo pasados como parametos de un método es mejor crear una clase que contenga todos esos parámetros.
  + Cuando los datos complicados se codifican en variables podemos usar el Remplazar Tipo de Código con una Clase o subclase.
  + Si hay matrices entre las variables podemos Remplazar la matriz con un objeto.

## Ejemplo: Netbeans

# 

# Nombre: Data Clumps – Grupo de Datos

## Descripción: Pertenece a los Bloaters y genera una mala practica en el código cuando en una clase hay muchos grupos de datos que no están relacionados entre sí , es decir hacer ver un código más complicado , por lo que deberían contar con su propia clase .

## Técnicas :

* + Si en una clase existe repetición de datos innecesarios podemos usar una clase extra para mover los campos a esa propia clase.
  + Si algunos de los datos se pasan a otros métodos podemos pasar todo el objeto que contiene a los datos al método en vez de pasar únicamente los campos.
  + Al igual que en el Primitive Obssesion si los mismos datos están siendo pasados en los parámetros de un método podemos usar Introduce Parameter Object para fijarlos como una clase.

## Ejemplo: Netbeans

# Nombre: Switch Statements – Cambiar Declaraciones

## Descripción: Pertenece a los Object-Orientation Abusers y representa un mal olor en el código debido a que comenzamos a usar muchos if o switch case para representar mas caminos a seguir dentro de nuestro programa según las circunstancias en el que se encuentre pero cuando deseamos agregar una nueva condición, entonces ,debemos encontrar todo el código del interruptor y modificarlo.

## Técnicas :

* + Podemos aislar el interruptor y colocarlo en una clase adecuada, se puede lograr mediante la extracción de métodos y luego moverlos.
  + Si un cambio se basa en el tipo del código , como cuando cambiamos el modo en tiempo de ejecución el programa , podremos usar Replace Type Code.
  + Si una de las opciones del condicional es nula , entonces debemos usar un Null Object.

## Ejemplo: Netbeans

# Nombre: Feature Envy – Caracteristica Envidiosa

## Descripción: Pertenece a los Couplers y genera un mal olor dentro del código después de que los campos se mueven a una clase de datos, ya que las operaciones existentes asociadas a esos datos serán parcial o incluso totalmente ignoradas , en este caso vamos a necesitar mover las operaciones en los datos a esta nueva clase también.

## Técnicas :

* + Si solo una parte de un método accede a los datos de otro objeto en este caso se podrá usar Extract Method.
  + Si un método emplea funciones de varias diferentes clases , primero evalue que clase contiene la mayoría de los datos utilizados y luego coloque el método en esta clase acompañada de los otros datos.
  + Si un método evidentemente debe ser trasladado hacia otro lugar use MoveMethod.

## Ejemplo: Netbeans

# Nombre: Large Class – Clase Larga

## Descripción: Pertenece a los Bloaters y genera un mal olor dentro del código después de cierto tiempo en el que la clase está llena de muchos campos, métodos y en general líneas de código , es decir , la clase tiene muchas responsabilidades que cumplir mientras lo ideal es que solo tenga una tarea en si .

## Técnicas :

* + Podemos usar Extract Class si el compartamiento de la clase grande se puede dividir en componentes por separado.
  + Si deseamos tener una lista de operaciones y compartamientos que el cliente podría emplear entonces podemos usar Extract Interface
  + Se puede emplear Extract Subclass en el caso de que parte del comportamiento de la clase grande se pueda implementar de diferentes maneras.

## Ejemplo: Netbeans

# Lista de referencias

* Kerievsky, Joshua (2004). Refactoring to Patterns. Addison Wesley.
* Suryanarayana, Girish (November 2014). Refactoring for Software Design Smells. Morgan Kaufmann. p.
* https://sourcemaking.com/refactoring
* https://refactoring.guru/refactoring/smells

# 