**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**DISEÑO DE SOFTWARE**

**SEGUDO PARCIAL**

**TRABAJO AUTÓNOMO**

Josue Cobos Salvador

ENERO 2020

Contents

[Malos olores escogidos 3](#_Toc29671338)

[1. Primitive Obsession 3](#_Toc29671339)

[2. Long Class 3](#_Toc29671340)

[3. Refused Bequest 3](#_Toc29671341)

[4. Switch statements 3](#_Toc29671342)

[5. Shotgun surgery 3](#_Toc29671343)

[Análisis 3](#_Toc29671344)

[Primitive Obsession (Obsesión de uso de primitivos) 3](#_Toc29671345)

[Lista de técnicas de refactorización : 3](#_Toc29671346)

[Long Class(Clase muy larga) 3](#_Toc29671347)

[Descripción corta 3](#_Toc29671348)

[Lista de técnicas de refactorización 3](#_Toc29671349)

[Refused Bequest(Herencia rechazada) 4](#_Toc29671350)

[Descripción corta 4](#_Toc29671351)

[Lista de técnicas de refactorización 4](#_Toc29671352)

[Switch statements(Sentencias de validación) 4](#_Toc29671353)

[Descripción corta 4](#_Toc29671354)

[Lista de técnicas de refactorización 4](#_Toc29671355)

[Shotgun surgery (Cirugía con escopeta) 4](#_Toc29671356)

[Descripción corta 4](#_Toc29671357)

[Lista de técnicas de refactorización 4](#_Toc29671358)

[Bibliografía 5](#_Toc29671359)

# Malos olores escogidos

## Primitive Obsession

## Long Class

## Refused Bequest

## Switch statements

## Shotgun surgery

# Análisis

### Primitive Obsession (Obsesión de uso de primitivos)

Descripción corta:   
Es un problema en el código ya que al manejar muchos subatributos de un atributo de una clase como primitivos, se dificulta la manipulación de estos y además se tendría que modificarlos uno a uno y no en conjunto como se debería.

### Lista de técnicas de refactorización :

1. Replace Data Value with Object .- Uso de pequeñas clases (Solo con atributos que necesiten ser manipulados en conjutno)
2. Introduce Parameter Object (en caso de que el valor de los campos primitivos son usados como parámetros de un método.
3. Replace Type Code with Class(Cuando los datos están codificados en las variables)
4. Replace Array with Object.(cuanto se tiene un arreglo de primitivos como atributo)

### Long Class(Clase muy larga)

### Descripción corta

Es un problema ya que al dejarle muchas responsabilidades a una sola clase, se complica mucho reutilizar las mismas clases en otro proyecto ni tampoco se podría extender tan fácilmente de ellas.

### Lista de técnicas de refactorización

1. Extract Class.- Extraer parte de las funciones o comportamiento de una clase sobrecargada y entregárselo a una nueva clase.(Cuando la nueva clase no comparte comportamiento con la original)
2. Extract Subclass.- Permite modelar cierto comportamiento que se da rara vez en una clase.(Cuando la nueva clase comparte comportamiento con la orginal)
3. Extract Interface.- Permite delegar cierto comportamiento a clases específicas.(Cuando la nueva clase no comparta atributos si no solo comportamiento)

### Refused Bequest(Herencia rechazada)

### Descripción corta

Es un problema porque al modelar incorrectamente la herencia, vamos a tener métodos que no se van a usar en ciertas clases que hereden de un superclase, haciendo que en la mayoría de los casos la herencia sea absurda.

### Lista de técnicas de refactorización

1. Replace Inheritance with Delegation.- En esta se reemplaza la herencia por una composición, asociación o agregación entre clases.

### 

### Switch statements(Sentencias de validación)

### Descripción corta

Es un problema ya que al llenar un método con sentencias if hacemos que el código sea muy poco legible, y difícil de entender.

### Lista de técnicas de refactorización

1. Extract Method.- Extraer parte del código de la sentencia if en otro método que maneje ciertas validaciones y no todas a la vez.(Cuando las validaciones se puedan dar por separado)
2. Replace Type Code with Subclasses.- Se usa herencia para que cada clase tenga su comportamiento y no exista la necesidad de hacer validaciones extras.(En el caso de que las validaciones sean por Clase)
3. Replace Conditional with Polymorphism.- Se deja que el polimorfismo nos ayude a implementar cierta funcionalidad dependiendo de la clase que llame el método.(En caso de que las validaciones sean dependientes a la clase que llama al método)
4. Replace Parameter with Explicit Methods .- Se basa en eliminar los métodos genéricos en los que se valide para manipular datos por métodos más particulares dependiendo de la funcionalidad.(En caso de que el bloque if o switch sean por generalizar un método para muchos casos).

### Shotgun surgery (Cirugía con escopeta)

### Descripción corta

Es un gran problema para el código, en especial para el mantenimiento o extensión, ya que al modificar un valor que se utiliza en muchos lugares del código se tendrá que cambiar uno a uno si no está bien implementado.

### Lista de técnicas de refactorización

1. Move Method.- Que se basa en mover un método donde en verdad se lo esté llamando más .(En el caso de que se esté llamando a un método desde otra clase diferente a la que llama el método )
2. Inline Class.- Busca eliminar clases redundantes agrupando las clases que tienen métodos que cumplen una misma responsabilidad.(Cuando el mismo método se llama desde más de una clase muy frecuentemente y las clases a agrupar guardan cierta relación o compartan responsabilidad)

# Bibliografía

Refactoring.guru. (2020). Refactoring: clean your code. [online] Available at: https://refactoring.guru/refactoring [Accessed 12 Jan. 2020].

Refactoring.guru. (2020). Refactoring: clean your code. [online] Available at: https://refactoring.guru/refactoring [Accessed 12 Jan. 2020].