

Refactoring

Semana 10

Agenda

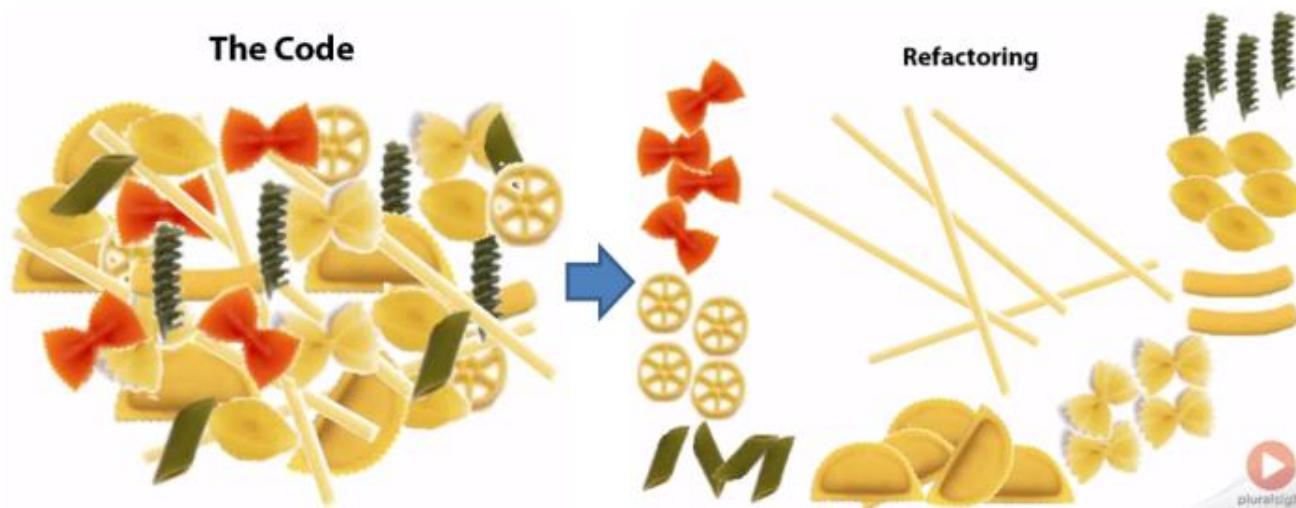
- Definición de refactorización
- Identificación de "malos olores" de programación.
- Clasificación de técnicas de refactorización.

Definición de refactorización



Refactorización

Es una transformación del código fuente sin alterar el comportamiento del sistema.



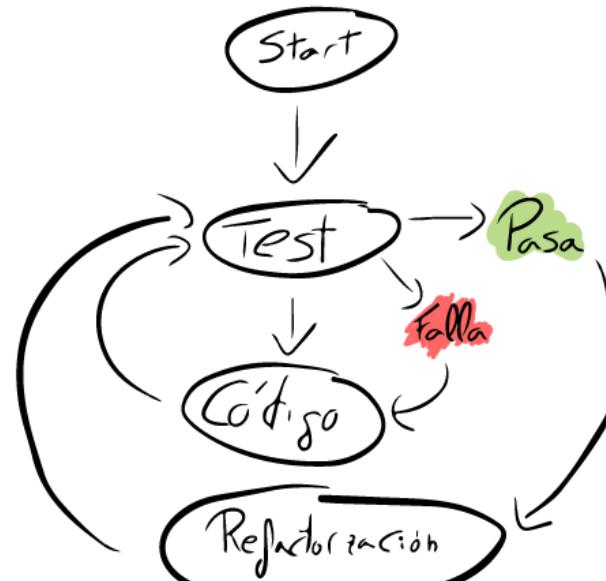
Refactorización

- Permite mejorar los siguientes aspectos:
 - Legibilidad
 - Mantenimiento del código (modificaciones)



Refactorización

- Es realizada en pequeños incrementos que son intercalados con pruebas.



Ventajas

- Hacer refactorización de código sistemáticamente, mantiene el código más sencillo y elegante .
- Hacer cambios es menos costoso.
- Depurar y entender el código tarda menos.
- Modificar y añadir código es más sencillo.

¿Durante la refactorización se puede añadir nueva funcionalidad al software desarrollado?

NO, su finalidad NO es añadir nuevas funcionalidades.

La refactorización se encarga de hacer más limpio el código, menos complejo, es decir, eliminar “malos olores” .

Refactoring: Ejemplo

BEFORE

```
class point(object):

    def __init__( self, x, y ):
        self.x, self.y = x, y

    def add( self, p ):
        x = self.x + p.x
        y = self.y + p.y
        return point( x, y )

    def sub( self, p ):
        x = self.x - p.x
        y = self.y - p.y
        return point( x, y )
```

AFTER

```
class Point(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, p):
        return Point(self.x + p.x, self.y + p.y)

    def __sub__(self, p):
        return Point(self.x - p.x, self.y - p.y)
```

<http://stackoverflow.com/questions/3152820/python-refactoring-similar-methods-in-class>

¿Tiene algún problema este código?

```
public double getTotalCost(){

    double quantityDiscount = 0.0;

    if((quantity > 50) || ((quantity * price) > 500)) {

        quantityDiscount = .10;

    } else if((quantity > 25) || ((quantity * price) > 100)) {

        quantityDiscount = .07;

    } else if((quantity >= 10) || ((quantity * price) > 50)) {

        quantityDiscount = .05;

    }

    double discount = ((quantity - 1) * quantityDiscount) * price;
```

¿Ahora es más entendible?

```
final boolean over50Products = (quantity > 50) || ((quantity * price) > 500);
final boolean over25Products = (quantity > 25) || ((quantity * price) > 100);
final boolean over10Products = (quantity >= 10) || ((quantity * price) > 50);

if(over50Products) {

    quantityDiscount = .10;

} else if(over25Products) {

    quantityDiscount = .07;

} else if(over10Products) {

    quantityDiscount = .05;

}

double discount = ((quantity - 1) * quantityDiscount) * price;
```

Videos con ejemplos

- CODE Refactoring – Derek Banas
- <https://www.youtube.com/watch?v=vhYK3pDUijk>

Identificación de "malos olores" de programación.

Malos olores

- ¿A qué nos referimos con un mal olor en el código?

```
a= DateTime.Now.Millisecond;
iber = new Random(seed);
: i= number.Next(0,9);

(maskedSet[i] ==0)

maskedSet[i]=1;
setCount++;
// Mask each set

seed = DateTime.Now.Millisecond;
number = new Random(seed);
int maskPos = number.Next(minPos,maxPos);
int j=0;
do
{
    seed = DateTime.Now.Millisecond;
    number = new Random(seed);
    int newPos = number.Next(1,9);
    int x = _setRowPosition[i]+pr
    int y= _setColPosition[i]+a
    if(_problemSet[x,y] ==
    {
        _problemSet[x,y] =
        j++;
    }
}
```



Malos olores

- Se refiere a cualquier síntoma en el código fuente de un programa que posiblemente indica un problema más profundo.
- Son problemas de diseño que se reflejan en el código



Malos olores: Síntomas

- Código duplicado (el mismo código repetido)
- Métodos largos (>10 LOC)
- Sentencias de condicionales complejas (dificultan el entendimiento)
- Clases vagas (casi no hacen nada, es mejor poner ese código en otra clase)
- Clases largas (clases con muchas responsabilidades)
- Explosión combinatorial (se usan muchos métodos en lugar de uno que puede hacer varias cosas)
- Oddball solution (solucionar un problema de varias maneras – se debe escoger la mejor solución para no duplicar código)

How can code “smell”?

Refactoring



Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

— *Martin Fowler* —

AZ QUOTES

Bloaters

- Métodos y clases que han alcanzado grandes proporciones.

BLOATERS

Long Method



Large Class

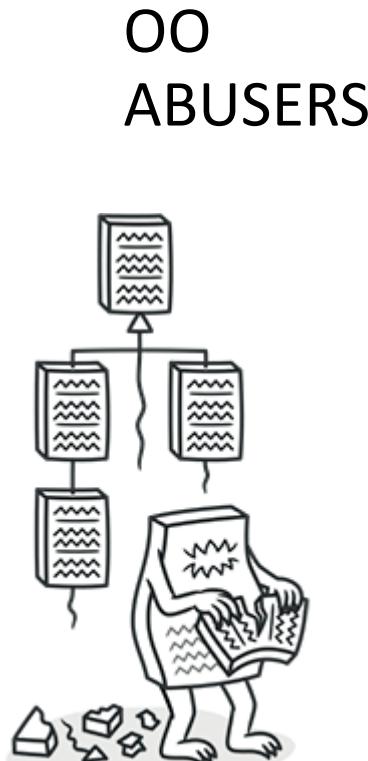
Primitive Obsession

Long Parameter List

Data Clumps

OO Abusers

- Resultados de una incorrecta aplicación de la orientación a objetos



Switch Statements

Temporary Fields

Refused Request

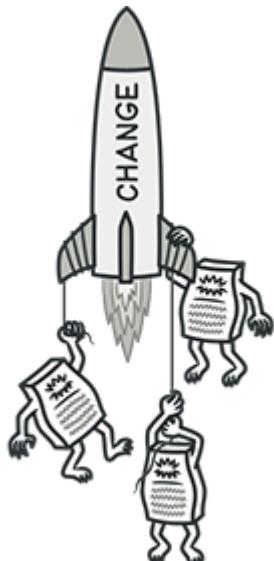
Alternative Classes with Different Interfaces

Change Preventers

- Si necesitas cambio en un lugar del código, pero resulta que debes cambiar en muchos otros lugares.

CHANGE
PREVENTERS

Divergent Change



Shotgun Surgery

Parallel Inheritance Hierarchies

Dispensables

- Código cuya ausencia haría que el resto del programa sea más limpio, eficiente y fácil de entender.

DISPENSABLES



Comments

Duplicate Code

Lazy Class

Data Class

Dead Code

Speculative Generaly

Couplers

- Código que contribuye a un acoplamiento excesivo entre clases.

COUPLERS

Feature Envy



Inappropriate Intimacy

Message Chains

Middle Man

Clasificación de técnicas de refactorización

Clasificación de técnicas de refactorización

1. Composing methods

- Streamline methods, remove code duplication, and pave the way for future improvements

2. Moving features between objects

- Move functionality between classes, create new classes, and hide implementation details from public access.

3. Organizing data

- Data handling, replacing primitives with rich class functionality, untangling of class associations

Clasificación de técnicas de refactorización

4. Simplifying conditional expressions
5. Simplifying method calls
 - Make method calls simpler and easier to understand
6. Dealing with generalisation
 - Moving functionality along the class inheritance hierarchy, creating new classes and interfaces, and replacing inheritance with delegation and vice versa

Antes de finalizar

Puntos para recordar

- ¿Qué es refactorización?
- ¿Qué son los malos olores?
- Recuerde un ejemplo de cada mal olor

Lectura adicional

- Source Making:
 - <https://sourcemaking.com/refactoring>
- Refactoring Guru:
 - <https://refactoring.guru/>

Próxima sesión

- Técnicas de Refactorización