Clase 14: Code Smells

Rodrigo Arturo Saffie Kattan

Pontificia Universidad Católica de Chile

rasaffie@ing.puc.cl

20 de septiembre de 2016

Contenidos

¿Qué es un Code Smell?

"A code smell is a surface indication that usually corresponds to a deeper problem in the system." [Fowler, M., 2006]

El término fue acuñado por Kent Beck, y se hizo famoso con el libro [Refactoring: Improving the Design of Existing Code, 2012].

No son bugs: el programa funciona correctamente, pero su débil diseño dificulta el desarrollo e incrementa la posibilidad de generar *bugs*.

¿Qué es un Code Smell?

Características

- Indicador superficial
- Rápido de detectar
- No siempre indican un problema en el código
- No es un *bug per se*

Clasificación

Existen 5 clasificaciones para los *code smells*:

- Bloaters
- Object-Orientation Abusers
- Change Preventers
- Dispensables
- Couplers

Bloaters

- *Bloaters* representan código, métodos y clases que han crecido a tal punto que es difícil trabajar con ellos.

- **Long Method**
- **Large Class**
- **Primitive Obsession**
- **Long Parameter List**
- **Data Clumps**

Object-Orientation Abusers

- Estos *code smells* se generan por una incompleta o incorrecta utilización de los principios OOP.

Switch Statements

Síntomas

- Se tiene un operador *switch* complejo, o una secuencia de *if*

Razones del problema

- Mal uso de polimorfismo

Beneficios

- Código más organizado

Temporary Field

Síntomas

- Se tienen atributos de una clase que se les asigna un valor bajo ciertas circunstancias. El resto del tiempo se encuentran vacíos

Razones del problema

- En vez de pasar datos como parámetros, se crean atributos para las clases

Beneficios

- Código más claro y organizado

Refused Bequest

Síntomas

- Una subclase utiliza solamente algunas de las propiedades y métodos heredados de sus padres

Razones del problema

- Se quiso reutilizar código entre una super-clase y una clase, pero son completamente distintas

Beneficios

- Código más claro y organizado

Alternative Classes with Different Interfaces

Síntomas

- Dos clases realizan funcionalidades idénticas, pero con nombres de métodos distintos

Razones del problema

- El programador que creo una clase probablemente no sabía que otra clase que ya existía tenía funcionalidad equivalente.

Beneficios

- Elimina código duplicado
- Código más claro y organizado

Referencias

Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D. (2012)
Refactoring: Improving the Design of Existing Code
1st ed., *Addison-Wesley Professional*

martinfowler.com
http://martinfowler.com/bliki/CodeSmell.html

https://refactoring.guru
https://refactoring.guru/smells/smells

Fin