Refactorización II

Fecha entrega: 23/03/15
Autor: Pedro J. Ramos
Estela Muñoz Cordón

1. **Identifica con los Bad smells ó Code smells descritos por Kent Beck. Algunos describen posibles técnicas de refactorización:**
    1. **Clases que sólo contienen datos. Hay que evitarlas. Las clases han de tener datos y comportamiento para operar sobre ellos.**

Data class. La existencia de estas clases debería de cuestionarse dado que no suelen tener comportamiento alguno.

    2. **Subclases que no usan todas las características de las superclases**

Refused bequest. Si las subclases no necesitan ni usan lo que obtienen mediante la herencia puede significar que la jerarquía de clases no es correcta. La delegación suele ser la solución.

    3. **Método que utiliza demasiada información de una clase ajena**

Feature envy. Se resuelve el problema pasando el método a la clase cuyos componentes son más requeridos para usarse.

    4. **Demasiados parámetros**

Long parameter list. Con la programación orientada objetos no se suelen utilizar muchos parámetros, la existencia de muchos parámetros hace que se haga difícil de comprender e incrementar el código.

    5. **Clase que soluciona muchos problemas. Normalmente se soluciona extrayendo clases o subclases para agrupar variables.**

Large class. Al intentar resolver muchos problemas tiene varias variables de instancia, lo que suele conducir a código duplicado.

6. **La misma expresión en distintos métodos de la misma clase. Habría que extraer un método e invocarlo en ambos sitios.**

Duplicated code. Motivo principal para refactorizar, se busca la forma de extraerlo y unificarlo.

7. **El cambio en un método desencadena muchos cambios en otras clases**

Shotgun surgery.

8. **Duplicación de la misma expresión en dos subclases hermanas. Habría que extraer un método y subirlo a la clase antecesora común a ambas.**

Problema de herencia, las clases definidas en las subclases hijas de forma idéntica debería formar parte de la clase padre y ser heredadas por estructura jerárquica.

2. Kent Beck. Indica quién es, así como sus trabajos.

Kent Beck es ingeniero de software estadounidense, uno de los creadores de las metodologías de desarrollo de software de programación extrema y el desarrollo guiado por pruebas, también llamados metodología ágil.

Fue pionero en patrones de diseño de software, el redescubrimiento del test-driven development, así como también de la aplicación comercial de Smalltalk. Con Ward Cunningham popularizó la metodología de tarjetas CRC, y con Erich Gamma el framework de pruebas unitarias para Java conocido como JUnit.