Patrones de refactorización más usuales

- **1. Duplicated code (código duplicado):** Es la principal razón para refactorizar. Si se detecta el mismo código en más de un lugar, se debe buscar la forma de extraerlo y unificarlo.
- **2. Long method (método largo).** Cuanto mas corto es un método más fácil de reutilizarlo es.
- **3. Large class (clase grande).** Si una clase intenta resolver muchos problemas, usualmente suele tener varias variables de instancia… lo que suele conducir a código duplicado.
- **4. Long parameter list (lista de parámetros extensa):** en la programación orientada a objetos no se suelen pasar muchos parámetros a los métodos, sino sólo aquellos mínimamente necesarios para que el objeto involucrado consiga lo necesario. Éste tipo de métodos, los que reciben muchos parámetros, suelen variar con frecuencia, se tornan difíciles de comprender e incrementan el acoplamiento.
- **5. Divergent change (cambio divergente):** una clase es frecuentemente modificada por diversos motivos, los cuales no suelen estar relacionados entre si.

Patrones de refactorización más usuales

- **6. Shotgun surgery (Cirugía de escopeta):** este síntoma se presenta cuando luego de un cambio en un determinado lugar, se deben realizar varias modificaciones adicionales en diversos lugares para compatibilizar dicho cambio.
- **7. Feature envy (envidia de funcionalidad):** un método que utiliza más cantidad de elementos de otra clase que de la propia. Se suele resolver el problema pasando el método a la clase cuyos componentes son más requeridos para usar.
- **8. Switch:** Lo tipico de un software Orientado a Objetos es el uso minimo de los "switch"(o case, o switch escondido). El problema es la duplicación, el mismo "switch" en lugares diferentes
- **9. Data class (clase de datos):** Clases que sólo tienen atributos y métodos de acceso a ellos ("get" y "set"). Este tipo de clases deberían cuestionarse dado que no suelen tener comportamiento alguno.
- **10. Refused bequest (legado rechazado):** Subclases que usan sólo pocas características de sus superclases. Si las subclases no necesitan o no requieren todo lo que sus superclases les proveen por herencia, esto suele indicar que como fue pensada la jerarquía de clases no es correcto.