Refactorings

La refactorización de código (del inglés refactoring) es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Esta característica que tienen los refactorings lo hacen muy útil para el mantenimiento de código porque nos permite transformar el código sin modificar el funcionamiento del sistema.

El mantenimiento de un sistema no es solo corrección de errores, sino que es una etapa en el ciclo de vida de desarrollo de un sistema. Según un estudio realizado por Meir M. Lehman más del 80% del esfuerzo de mantenimiento es usado para acciones *no correctivas*, sino que son mejoras en la funcionalidad del sistema: agregando nuevos *features* o haciendo mejoras de usabilidad y optimización, etc.

Entre las conclusiones de la investigación de Lehman se incluye que el mantenimiento es realmente un desarrollo evolutivo y que las decisiones de mantenimiento son ayudadas por entender lo que sucede a los sistemas (y al software) con el tiempo. Lehman demostró que los sistemas continúan evolucionando con el tiempo. A medida que evolucionan, los sistemas se hacen más complejos a menos que se hagan refactorizaciones de código con cierta frecuencia para reducir la complejidad.[1]

La capacidad de hacer refactorings está dada por las herramientas que nos provea el entorno de programación. Si el entorno de programación no provee la posibilidad de hacer refactorings o solo algunos pocos deberíamos hacerlos manualmente, lo que podría introducir errores por olvidarnos de modificar en alguna parte del sistema.

[1] Lehman, Meir M. (1980). «Programs, Life Cycles, and Laws of Software Evolution». Proc. IEEE 68 (9): 1060-1076.

Para poder realizar los refactorings de forma segura, ya sea automáticamente con la herramienta que nos provea el entorno de desarrollo o manualmente, es imprescindible tener una buena cobertura del código con tests. De esta manera, antes de aplicar el refactoring corremos los tests, si estos pasan, aplicamos el refactoring y volvemos a correr los tests. Si luego de aplicar el refactoring, no pasan corregimos lo que nos haya quedado mal. Teóricamente deberían pasar porque el refactoring no cambia el comportamiento del sistema, pero puede ser que los tests estén muy acoplados al código o que hayamos hecho el refactoring manualmente y nos haya quedado algún error, etc. Entonces los tests nos dan esa certeza de que el sistema sigue haciendo lo que estaba haciendo.

A continuación describiremos algunos de los tipos de refactorings de código más utilizados:

Refactoring	**Descripción**
Rename	● Renombra el elemento seleccionado y todas las referencias.
Extract Method	● Crea un nuevo método con el código seleccionado y lo reemplaza por un envío de mensaje a ese método. ● Algunas herramientas reemplazan todas las repeticiones de la selección.
Extract to Local - Introduce Variable	● Extrae el código seleccionado en una variable local inicializada con dicho código.
Move Method	● Mueve el método seleccionado a una clase que debe estar visible desde el contexto en que se mueve. ● Modifica todas las referencias al método movido.
Convert local to field - Introduce Field	● Convierte una variable local en variable de instancia. ● Puede mover la inicialización de la variable.

Introduce a parameter	● Reemplaza el código seleccionado con una referencia a un parámetro , modifica todos los senders agregando dicho parámetro que será el código reemplazado.
Change Method Signature - Change Signature	● Permite modificar los elementos que definen un método como los parámetros. ● Modifica todos los senders para quedar de acuerdo con la nueva definición. ● Cuando se agregan parámetros se puede indicar con qué código lo llaman los senders.
Inline	● Copia el código representado por el método o variable a los lugares donde se referencia dicho método o variable.
Encapsulated field	● Referencias a variables de instancia se reemplazan por getters y setters.
Extract Parameter Object - Extract Class from Parameters	● Permite crear una abstracción nueva a partir de un conjunto de parámetros de un método. ● Reemplaza todos los senders por la instanciación de la nueva abstracción.
Pull Up	● Mueve variables de instancia y/o métodos a la superclase o declara el método como abstracto en la superclase.
Push Down	● Mueve un conjunto de variables de instancia y/o métodos a las subclases.