Refactoring Ejemplo 2

Por: Juan Sebastián Sánchez L

Refactoring: Extracción de función (Extract Function)

- Descripción: Se debe tomar un fragmento de código y encapsularlo en una función independiente. En este caso, se ha extraído la funcionalidad de organizar los logs en la función organizar.
- Razón de aplicación: Se aplica para dividir el código en partes más pequeñas y manejables, lo que mejora la legibilidad y el mantenimiento al dar a cada función una única responsabilidad.
- Consecuencia: El código principal se vuelve más legible, más claro y más mantenible, ya que se separa la lógica específica de organizar los logs en una función independiente.

Refactoring: Extracción de variables

- Descripción: Implica tomar valores literales o expresiones complejas y asignarlos a variables con nombres descriptivos. En este caso, se ha extraído la variable *error_file_path*.
- Razón de aplicación: Se aplica para evitar la repetición innecesaria de valores en el código, lo que mejora la claridad y evita posibles errores tipográficos.
- Consecuencia: El código se vuelve más claro y menos propenso a errores, ya que se utilizan variables con nombres descriptivos en lugar de valores literales.

Refactoring: Alineación de código

- Descripción: La alineación de código implica formatear el código de manera que las estructuras (como tabulaciones, espacios y paréntesis) estén alineadas de manera consistente en todo el programa.
- Razón de aplicación: Se aplica para mejorar la legibilidad y la coherencia del código, facilitando la identificación de patrones y mejorando el mantenimiento.
- Consecuencia: El código se vuelve más fácil de leer y entender, ya que sigue un formato uniforme, lo que hace que sea más sencillo de mantener y depurar.

Refactoring: Reemplazo de Asignación con Acumulación

- Descripción: Implica reemplazar la inicialización de una lista vacía (en este caso, errores) y luego acumular valores en esa lista dentro de un bucle por una lista ya existente (logs) en la que se acumulan los valores directamente en ese bucle.
- Razón de aplicación: La razón para aplicar este refactoring es simplificar el código y hacerlo más eficiente. Al usar una lista existente (logs) directamente, se reduce la complejidad y se mejora la legibilidad del código.
- Consecuencia: La consecuencia principal de este refactoring es una simplificación del código, ya que se elimina la necesidad de inicializar una lista separada (errores) y, en su lugar, se acumulan los valores en la lista existente (logs). Esto reduce la cantidad de código y hace que el propósito del bucle sea más claro, lo que mejora la calidad y mantenibilidad del programa. Además, este cambio puede ser más eficiente en términos de uso de memoria al evitar la creación de una lista adicional.