Metodología de la programación II
Practica X

Emiliano Salvatori

Octubre 2019

1. Practica X

Refactorización

Responda las siguientes preguntas e indique las fuentes de información que utilizó.

¿Qué es un código simple? ¿qué características tiene?

Las características de un código simple son las siguientes:

- Funciona bien.
- Comunica bien lo que esta haciendo.
- No tiene duplicación.
- Tiene el menor número posible de clases y método.

Y a su vez proporciona los siguientes beneficios:

- El código es más fácil de cambiar, evolucionar o arreglar.
- El código es más fácil de leer y entender.
- Es más fácil hacerlo bien desde la primera vez, así estamos programando más rápido.

¿Qué significa *refactorizar*?

Refactorizar significa cambiar el código internamente sin alterar su funcionalidad externa. En general, con motivos de mejorar el diseño y obtener un código más simple.

¿En qué consiste la refactorización?

La Refactorización enseña técnicas para descubrir el código de mala calidad y técnicas para cambiarlo.
En el proceso es importante contar con un buen lote de casos de prueba que validen el correcto funcionamiento del sistema.
Los pasos a seguir son los siguientes:

- Ejecutar las pruebas, para obtener información sobre el comportamiento actual del sistema.
- Analizar los cambios a realizar.
- Aplicar los cambios.
- Ejecutar las pruebas y corroborar que los resultados antes y luego de efectuada la refactorización son iguales.

¿Qué diferencia hay entre optimizar y refactorizar?

La diferencia es que:

- En ambos casos se modifica el código fuente sin alterar el comportamiento observable del software.
- En la optimización se le suele agregar complejidad al código.

¿Cuáles son las ventajas que ofrece el proceso de refactorización?

Los aspectos favorables son:

- Favorece el mantenimiento del diseño del sistema.
- Facilita la lectura y comprensión del código fuente.
- Facilita la detección temprana de errores.
- Permite programar mas rápido, lo que eleva la productividad de los desarrolladores.

¿En qué situaciones se debe refactorizar y cuándo no tiene sentido hacerlo?

Hay que refactorizar cuando:

- Estamos agregando nueva funcionalidad al código.
- Estamos solucionando una falla.
- Estamos haciendo revisión de código (Distribución del conocimiento dentro del equipo de desarrollo).

Se debe evitar hacerlo cuando:

- Es más fácil hacerlo de nuevo.
- El código no funciona.
- Demasiado cerca de la fecha de entrega comprometida.

¿Qué problemas ocasionan los Métodos Largos? ¿qué deberíamos de hacer para eliminarlos?

Los Métodos Largos en el código, suelen tener las siguientes defectos:

- Dificultan mucho su comprensión.
- Seguramente realizan más de una responsabilidad.
- En la POO cuando más corto es un método más fácil su reutilización.
- Programas con métodos mas cortos, tienen vida mas larga.

Para eliminarlos se sugiere detectar las diferentes responsabilidades y sacarlas a métodos o clases nuevas.

¿A qué nos referimos cuando hablamos de "Envidia de Funcionalidades"?

Lo que se denomina "Envidia de Funcionalidades" contiene contiene las siguientes características:

- Métodos de una clase más interesados en datos de otra clase que en los datos suyos.
- La envidia de la Clase A por recursos de la Clase B es una indicación del acoplamiento fuerte de la Clase A con la Clase B.

Las posibles soluciones a esto son:

- Mover la funcionalidad del método de Clase A a la Clase B, que ya está más cerca de la mayoría de datos implicados en la tarea.
- Extraer a un método el código envidioso y mover sólo ese método a la Clase B.

¿Qué significa "Legado rechazado"? ¿Cómo podemos resolver este problema?

Es cuando el código en la POO tiene las siguientes características:

- 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.

La posible solución es utilizar la *Delegación* para subsanar los posibles errores de diseño.

Describa la técnica "Extraer Método". Dé un ejemplo donde se utilice dicha técnica

En la técnica denominada "Extraer método", los pasos a seguir son los siguientes:

- Se tiene un fragmento de código que es posible agrupar:

```
void imprimirDebe() {
        imprimirEncabezado(); //print details
        Console.Out.WriteLine("Nombre:" + nombre);
        Console.Out.WriteLine("Monto:" + debe());}
```

- Se transforma el fragmento a un método nuevo cuyo nombre va a explicar su propósito:

```
void imprimirDebe() {
        imprimirEncabezado();
        imprimirDetalle(debe()); }
void imprimirDetalle(double valor){
         Console.Out.WriteLine("Nombre:" + nombre);
        Console.Out.WriteLine("Monto:" + valor);}
```