Tecnologías de Programación

Paradigma Orientado a Objetos

VI – AntiPatrones de Diseño

UNL FICH

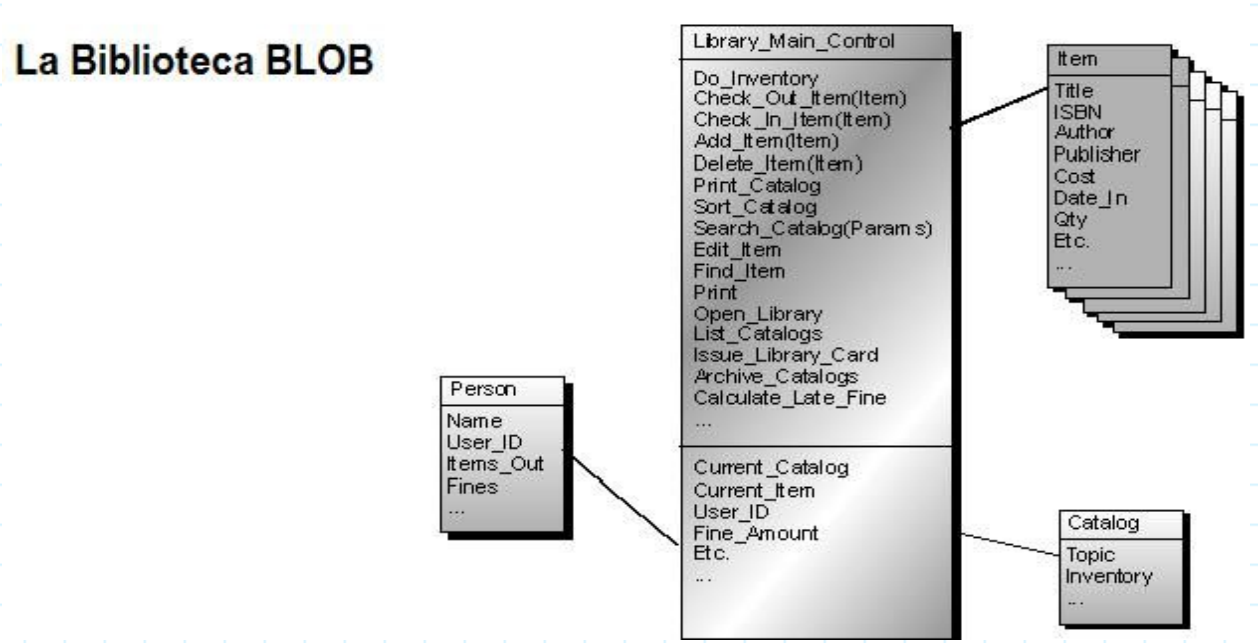
Ing. Pablo Garello

“ Aprendiendo de los errores de otros ”

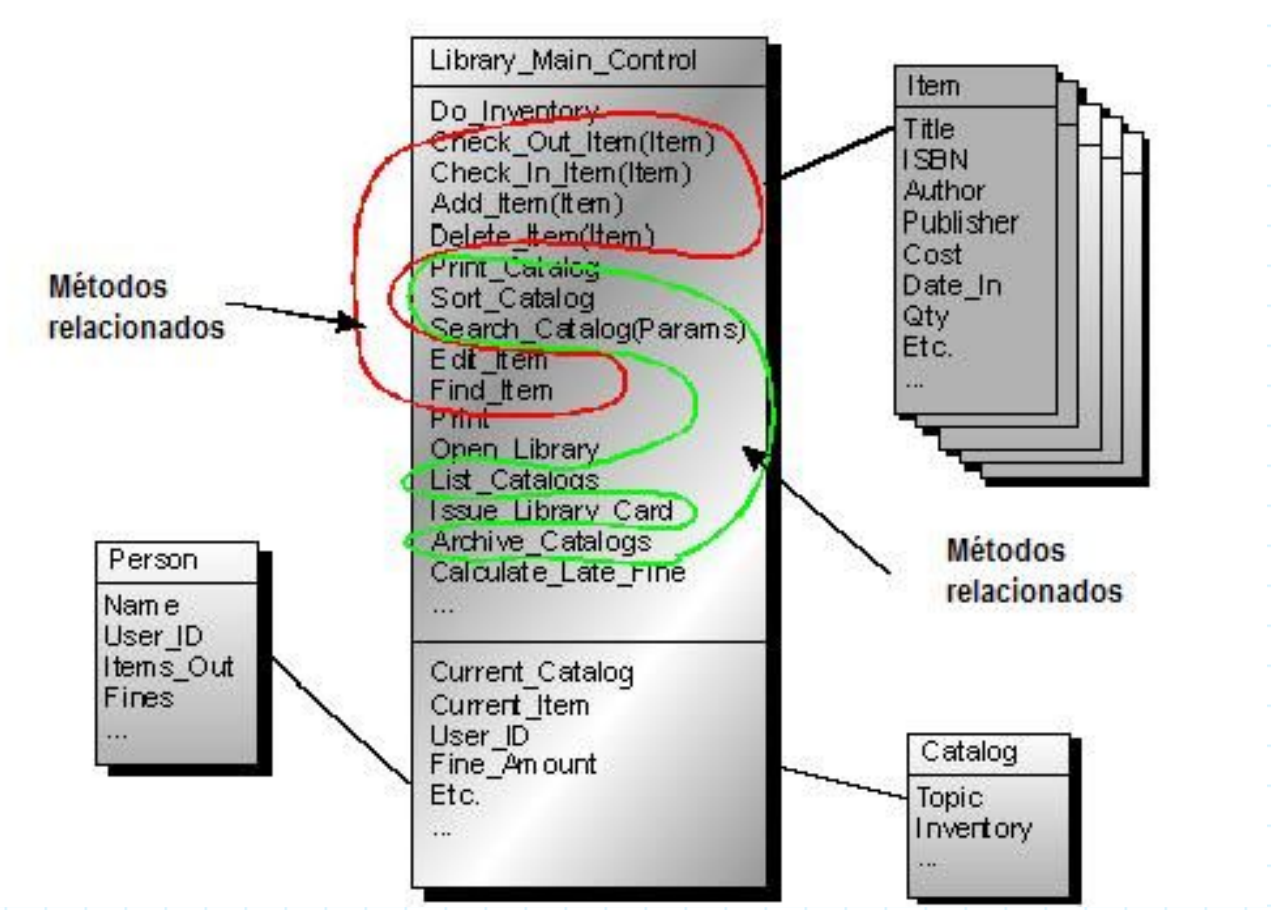
- Los patrones nos ofrecen una forma de resolver un problema típico, los ***antipatrones*** nos enseñan formas de enfrentarse a problemas con consecuencias negativas conocidas.
- Los ***antipatrones*** se basan en la idea de que puede resultar más fácil detectar “a priori” fallos en el desarrollo del proyecto que elegir el camino correcto, o lo que es lo mismo, descartar las alternativas incorrectas nos puede ayudar a la elección de la mejor alternativa.

Objeto Todopoderoso

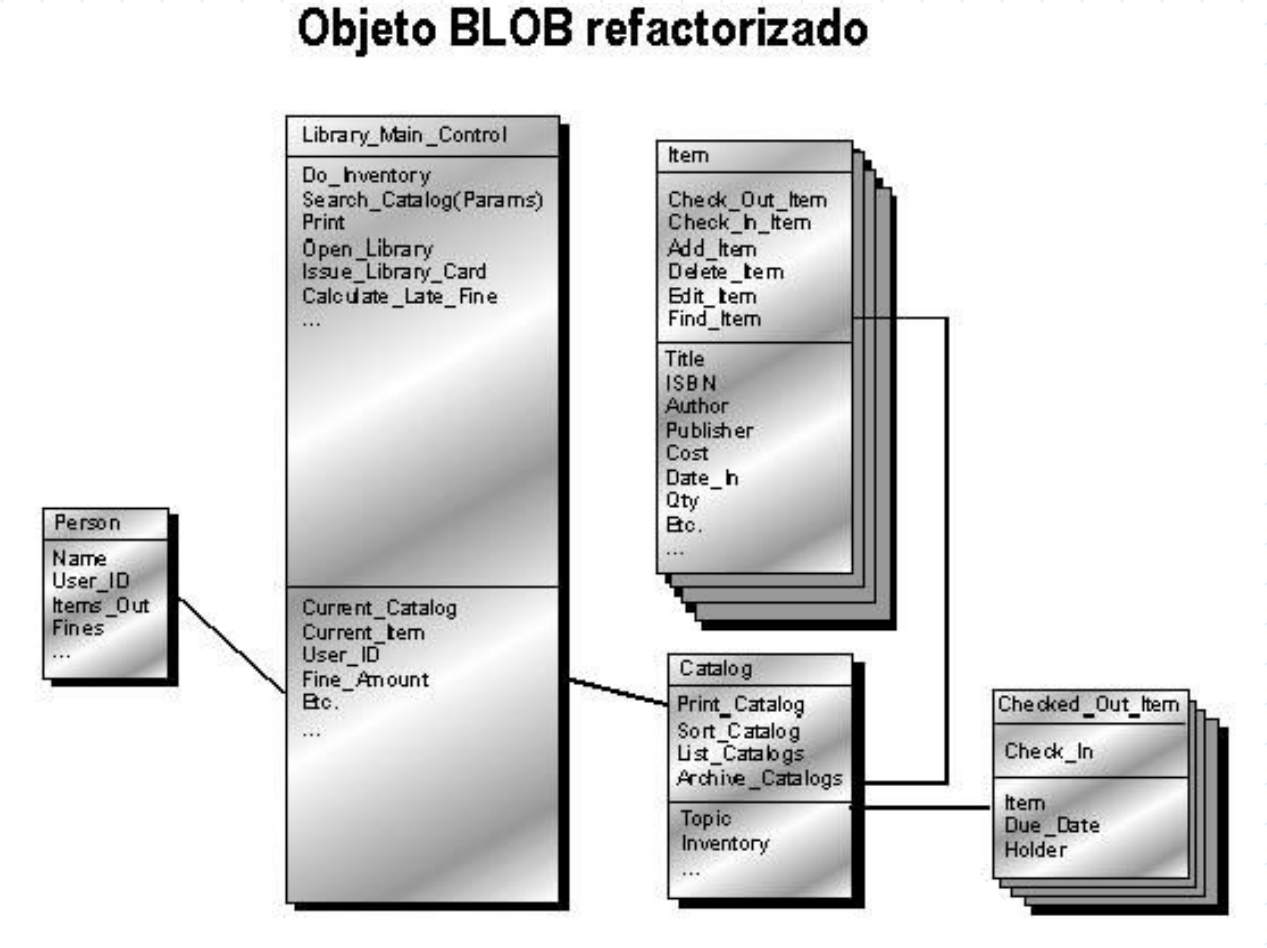
- Este **antipatrón** se da en objetos con muchos atributos o muchas operaciones. Esto rompe las ventajas de la programación OO, ya que estas clases tan grandes son muy difíciles de mantener, de re-usar y de probar. Suele aparecer por un diseño malo o debido a sistemas heredados. BLOB (Binary Large Objects)



- Los métodos y atributos relacionados deben situarse en sus lugares naturales.



- Aplico el patrón Refactory



Código espagueti

- El código spaghetti es un término peyorativo para los programas de computación que tienen una estructura de control de flujo compleja e incomprensible. Su nombre deriva del hecho que este tipo de código parece asemejarse a un plato de espaguetis, es decir, un montón de hilos intrincados y anudados.

UNL FICH

Hard Code

- Esta práctica se refiere a incrustar datos directamente en el código de una aplicación que deberían ser obtenidos de archivo, de la línea de comandos etc. Esto hace que cualquier cambio implique la modificación del código fuente. El ejemplo más habitual es fijar en el código el path de un fichero, si este cambia el programa no funciona. Lo correcto sería obtener la ruta de un archivo de configuración.

Acoplamiento Secuencial

- Construir una clase que necesita que sus métodos se invoquen en un orden determinado para poder realizar el proceso en forma correcta.
- Se resuelve aplicando el patrón de diseño TEMPLATE METHOD (Plantilla)

Copiar y Pegar

La programación mediante cortar y pegar es muy común, pero es una forma de degenerada de reutilización de software la cual crea pesadillas a la hora del mantenimiento. Proviene de la idea de que es más fácil modificar código existente que hacerlo desde cero. Esto es generalmente cierto y representa un buen instinto de software. Sin embargo, la técnica puede ser usada de manera excesiva.

Causas:
- Se necesita un gran esfuerzo para crear código reutilizable y las organizaciones hacen más hincapié en ganancias a corto plazo que en inversión a largo plazo.
- La organización no promueve componentes reusables y la velocidad de desarrollo eclipsa todos los demás factores de evaluación.

Reinventar la Rueda

- Consiste en intentar solucionar un problema conocido con una solución “ad hoc”, inventada y desarrollada por nosotros y sin tener en cuenta los patrones de diseño, arquitectónicos ni de ningún tipo. La causa principal suele ser el desconocimiento y la soberbia.

Recalculando ...

Desarrollar software reutilizable es más difícil y costoso que no reutilizable.

- Implica resolver una familia de problemas en lugar de resolver un único problema particular.
- Implicar saber programar software flexible y prever los ejes de cambio

Reutilización

Conviene diseñar e implementar elementos de software que puedan usarse para construir muchos programas diferentes ...

- Ahorra esfuerzo (evitamos “reinventar la rueda”)
- Permite concentrarse siempre en funcionalidad nueva
- Mejora indirectamente la calidad

UNL FICH