Anti Pattern

dalam Pengembangan Sistem Berorientasi Obyek

Pengertian Anti Pattern

- Design Pattern yang mengakibatkan pengenmbangan menjadi *counterproductive* dan tidak efisien (Koenig, 1995)
- Harus memenuhi 2 elemen utama:
  - Penerapan berulang dari pola perancangan, yang lama-lama menjadi tidak efektif lagi
  - Solusi yang lebih efektif telah tersedia

Domain Anti Pattern

- Organizational AP
- Project Management AP
- Programming AP
- Software Analysis AP
- Object-Oriented Design AP
- Methodological AP

Object-Oriented Design AP

- Merupakan Anti-pattern yg muncul karena kesalahan/ketidakefisienan menerapkan fitur atau pola perancangan berbasis obyek
- Adanya fitur/pola perancangan berbasis obyek malah membuat sistem kurang efisien
- Bisa juga terjadi karena *overuse* dari sebuah pola desain

Object Oriented Design AP

- Beberapa AP pada OOP:
  - BaseBean
  - CallSuper
  - Circular Dependency
  - God Object
  - Inappropriate Object Pool
  - Object Orgy
  - Poltergeist
  - Sequential Coupling
  - Yo-yo Problem

Call Super

- Symptoms:
  - Subclass are required to call methods from the super class
- In-appropriate use of common design patterns: inheritance
- Solution:

Circular Dependency

- Dimana dua objek saling membutuhkan satu dengan yang lain, e.g.,:
  - Class A membutuhkan class B, class B tergantung pada A
- Dapat menimbulkan efek berantai
- Dapat menimbulkan *memory leak*
- Biasanya ditimbulkan adanya *forward-declaration*
- Solusi: gunakan agregasi/inheritance dengan tepat

Constant Interface AP

- Muncul pada Bahasa Pemrograman Java
- Menggunakan *interface* untuk mendefinisikan *constants*
- menimbulkan polusi *namespace*

God Object AP

- Object yang menyimpan terlalu banyak data, dan melakukan terlalu banyak fungsionalitas
- Dalam OOP, pemisahan dan modularitas (prinsip *separation of concerns)* sangat penting
- Penerapan prinsip ini dengan tepat akan menghindari adanya God Object
- Solusi: Hierarchy kelas yang baik, komposisi, modularisasi

Object Pool

- Membuat Pre-inisialisasi dari beberapa object yang mungkin diperlukan
- Jika terlalu sering diterapkan, akan counterproductive (kebutuhan memory)
- Analisa yang tepat mengenai penggunaan pre-inisialisasi objek

Sequential Coupling

- Kelas yang memiliki methods, yang harus dipanggil melalui urutan tertentu
- Jika urutan tidak terpenuhi, muncul error/undefined behavior
- Desain yang baik menghindari adanya keharusan memanggil dengan urutan tertentu

Yo-yo Problem

- Terlalu banyak kelas/inheritance
- Untuk memahami cara kerja sistem, programmer harus mengikuti alur yang panjang
- Perpindahan informasi pada berbagai kelas
- Solusi: inheritance yang tidak terlalu tinggi