**DESIGN PATTERN**

Permasalahan dalam membuat sebuah perangkat lunak dari sejak awal sejarah perkembangan komputer selalu ada. Permasalahan tersebut dapat berupa hal yang bersifat teknis berhubungan langsung dengan arsitektur perangkat lunak yang dipakai dan juga bersifat perancangan yang lebih bersifat umum. Kedua hal tersebut adalah hal yang wajib dalam pembuatan sebuah perangkat lunak.

Bagaimanakah cara untuk mengelola spesifikasi yang ada sekarang secara efisien untuk kemudian mampu untuk menghadapi perubahan dan spesifikasi yang akan datang dalam kontek pembuatan dan pengembangan sebuah sistem adalah sebuah permasalahan lain.

Hambatan-hambatan yang sering dijumpai dalam pengembangan perangkat lunak tersebut antara lain.

1. kebutuhan yang akan datang sering tidak bisa dipahami dan merupakan sebuah hal yang sering sekali berubah.
2. usaha untuk menggunakan kembali (reuse) sering kali berujung melibatkan komponen-komponen yang sebenarnya tidak bisa digunakan kembali (not reusable) dan sering kali gagal untuk dilakukan.
3. sering sekali terjadi, programer dan sistem analis kehilangan gambaran secara umum tentang apa yang dimaksud dengan kebutuhan yang akan datang.
4. dan yang paling penting adalah para pengembang cenderung melihat permasalahan ke arah permasalahan coding. Pengembang gagal melihat bahwa penyelesaian masalah melalui kekuatan perancangan/desain adalah sesuatu yang patut dipertimbangkan dan merupakan jalur yang tepat.

**DESIGN PATTERN**

Adalah sebuah istilah dalam [rekayasa perangkat lunak](http://id.wikipedia.org/wiki/Rekayasa_perangkat_lunak" \o "Rekayasa perangkat lunak) yang mengacu kepada solusi umum yang dapat digunakan secara berulang kali untuk menyelesaikan masalah-masalah umum yang ditemukan dalam desain [perangkat lunak](http://id.wikipedia.org/wiki/Perangkat_lunak" \o "Perangkat lunak). Sebuah pola desain tidak berbentuk solusi akhir yang dapat langsung diterjemahkan menjadi [kode program](http://id.wikipedia.org/wiki/Kode_program" \o "Kode program).

Pola desain merupakan penjelasan atau templat yang menunjukkan bagaimana cara menyelesaikan sebuah masalah yang kemudian dapat digunakan di berbagai situasi yang berbeda-beda. Pola desain untuk [*object-oriented*](http://id.wikipedia.org/w/index.php?title=Object-oriented&action=edit&redlink=1) biasanya menunjukkan relasi dan interaksi antar kelas dan [objek](http://id.wikipedia.org/wiki/Objek" \o "Objek), tanpa menjelaskan kelas dan objek akhir yang terlibat dalam sebuah [aplikasi](http://id.wikipedia.org/wiki/Aplikasi" \o "Aplikasi). [Algoritma](http://id.wikipedia.org/wiki/Algoritma" \o "Algoritma) biasanya tidak disebut sebagai pola desain, karena algoritma menjadi solusi masalah [komputasi](http://id.wikipedia.org/wiki/Komputasi" \o "Komputasi) bukan masalah desain test.

Design Pattern adalah suatu solusi yang umum dilakukan untuk menangani masalah perancangan software.Design Pattern yang cukup populer diperkenalkan oleh GOF(Gang Of Four). Dewanya Design diantaranya adalah Erich Gamma, Richard Helm, Ralph Johnson dan John Vlissides.Dalam penjelasan dari Gang Of Four(GoF) terdapat 23 Pattern yang di bagi menjadi 3 kelompok besar. yakni diantaranya

1. Creation Patterns(cara Class/object di inisialisasi)
2. Structural Patterns (Struktur/ relasi antar object/class)
3. Behavior Patterns (Tingkah laku atau fungsi dari class/object)

Penjelasan secara singkat bagian-bagian Design Pattern :

1. *Creation Pattern*, yaitu pattern yang menyangkut dengan pembuatan object.Pattern akan menangani pembuatan suatu object, daripada kita menangani pembuatan objcet   
   secara langsung dan mungkin akan tersebar di dalam code kita. Dengan cara ini program akan lebih fleksibel dalam memutuskan pemakaian object yang dibutuhkan.
2. *Structural Pattern*. yaitu pattern yang menyangkut dengan struktur program. Dimana dalam Pattern ini akan lebih konsen ke class objcet composite. akan banyak penggunaan pewarisan to menggabungkan interface dan menjelaskan cara untuk menggabungkan object tujuan membuat fungsionalitas baru.
3. *Behavioural Pattern*. yaitu pattern yang menyangkut tentang kelakuan program.Dimana pada pattern ini akan menjelaskan spesifik tentang komunikasi antar objcet.

Dari ketiga kelompok besar design pattern tersebut memiliki bagian2 seperti berikut:

**Creational Pattern**

1. Abstract Factory (Pembuatan Instance Object dari keluarga object(Children Object))
2. Builder (Kontruksi Object yang kompleks dengan memisahkan kontruksinya dan penjabaran object tersebut)
3. Factory Method (Pembuatan Objet tanpa spesifikasi penjabaran class yang akan di buat)
4. Prototype ( Pembuatan object dengan mengkloning dari object yang telah ada)
5. Singleton ( Pembatasan pembuatan object dari suatu class)

**Structural Pattern**

1. Adapter (Pattern ini Membolehkan class yang interfacenya tidak compatible untuk bekerja bersama, dengan cara membungkusnya menjadi interface dari class yang telah ada)
2. Bridge ( Pattern ini Memisahkan abstraksi dari implementasinya sehingga dapat memberikan variasi class secara independent)
3. Composite (Pattern ini Menggabungkan satu atau lebih object yang serupa sehingga dapat dijadikan satu object)
4. Decorator (Pattern yang memberikan cara yang dinamis untuk menambahkan dan meng-override prilaku dari method yang ada dalam sebuah object)
5. Facade (Pattern yang menyediakan interface sederhana yang akan di implementasikan dengan coding yang besar)
6. Flyweight ( Pattern ini berupaya untuk mengurangi pembuatan dan manipulasi dari sejumlah object besar yang serupa)
7. Proxy (Pattern yang menyediakan tempat untuk objek-object mengakases controll dari aplikasi, dimana dapat mengurangi biaya dan mengurangi kompleksitas)

**Behavior Patterns**

1. Chain of responsibility ( Pattern yang mendelegasikan perintah untuk rantai pengolahan objek .
2. Command ( Menciptakan objects yang dikemas dengan beberapa pilihan/action dan parameter )
3. Interpreter ( Pattern yang digunakan untuk spesifik bahasa )
4. Iterator ( Mengakses element dari object secara berurutan tanpa memaparkan representasinya )
5. Mediator ( Memberikan pola untuk 'loose coupling' antara class dengan cara menjadikanya satu kelas yang mempaparkan detail dari method).
6. Memento ( Tanpa melanggar konsep enkapsulasi. Merekam dan mengektrasi keadaaan internal object dimana memungkinkan mengembalikan keadaannya ke semula nanti.)
7. Observer ( Mendefinisikan dependency one-to-many antara object, dimana perubahan terjadi pada satu object yang nantinya akan mengenai object turunannya secara otomatis )
8. State ( Memungkinkan object untuk mengubah prilaku ketika terjadi perubahan internal. )
9. Strategy (Menentukan sebuah keluarga algoritma, merangkumnya dan membuat mereka bisa saling di pertukarkan. Pattern ini memberikan algoritma yang bervariasi secara independent bisa digunakan)
10. Template Method ( Mendefinisikan kerangka operasi algoritma dalam sebuah abstract class. Metode template ini memungkinkan subclass mendefinisikan langkah tertentu dari sebuah algoritma tanpa mengubah struktur algoritma)
11. Visitor ( Memisahkan algoritma dari struktur object dengan memindahkan hirarki dari methods ke dalam satu object) Keuntungan dari Design Pattern adalah:  
    a. Design yang berulang, lebih effisien dari hanya penggunaan code yang berulang.  
    b. Menangani permasalahan yang umum dihadapi oleh system designer.  
    c. Dengan menggunakan Design Pattern akan lebih mudah bagi orang lain memahami apa yang anda lakukan dan mengapa.
12. Design Patter membantu anda untuk refactoring code anda dengan mudah

**TUGAS SEMESTER PENDEK POLA POLA PERANCANGAN**

**RANGKUMAN MATERI POLA POLA PERANCANGAN**

**PENGENALAN DESAIGN PATTERN**



**Disusun oleh:**

(AGUNG SETYAWAN)

(0910651211)

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MUHAMMADIYAH JEMBER**

**2014**