JURNAL DASI ISSN: 1411-3201
Vol. 11 No. 4 DESEMBER 2010

PENERAPAN DESIGN PATTERNS UNTUK PERANCANGAN BERBASISKAN OBJEK ORIENTED

Kusnawi
Dosen STMIK AMIKOM Yogyakarta

Abstraksi

Dalam mendesain suatu perangkat lunak cara penyelesainya biasanya didasari dari pemahaman secara pribadi atau bersifat subjektif sehingga dibutuhkan gambaran secara formal dari suatu masalah dan berikut pemecahannya. Design patterns adalah unsur-unsur rancangan yang seringkali muncul pada berbagai sistem yang berbeda. Setiap kemunculan ini menguji pattern tersebut di berbagai situasi. Design pattern harus mempunyai nama yang sederhana dan deskriptif yang dapat langsung digunakan untuk mengacu pada pola tersebut. Sebuah design pattern harus mendokumentasikan permasalahan, pemecahan, serta akibat-akibat penggunaannya. Class diagram adalah salah satu bentuk dari interprestasi dari suatu pattern dengan memanfaatkan kemampuan UML yang sudah berorentasi pada perancangan yang berbasiskan objek (OOP).

***Kata kunci** : Design Patterns, Object Oriented Programming, UML.*

Pendahuluan

Seorang software arsitek biasanya selalu menerapkan solusi yang potensial ketika memecahkan masalah yang terjadi pada saat pengembangan software. Solusi awal yang diterapkan atas masalah-masalah yang terjadi tergantung kepada pertimbangan masing-masing arsitek yang bersifat subjektif, sehingga dimungkinkan solusi yang ditawarkan belum tentu optimal, yang pada saat tertentu harus memperbaiki solusi yang lebih efektif pada kesempatan berikutnya.

Dalam pengembangan software sering terjadi permasalahan yang terjadi berulang-ulang, sehingga seorang arsitek mungkin akan banyak menghabiskan waktu dalam upaya untuk menerapkan solusi dengan masalah yang serupa.

Pada saat merancang software, patterns adalah suatu dokumen yang sangat penting untuk dimiliki dan dipahami. Dimana design

patterns bukan sekedar menyediakan solusi terbaik dalam memecahkan suatu masalah, tetapi lebih dari pada itu patterns membuat design software menjadi lebih efektif, fleksible, resilient dan waktu penyelesaikan desain software juga lebih efisien.

Design Patterns

Design patterns adalah unsur-unsur rancangan yang seringkali muncul pada berbagai sistem yang berbeda. Setiap kemunculan ini menguji pattern tersebut di berbagai situasi. Semakin terujinya suatu unsur rancangan berarti semakin matangnya unsur tersebut sehingga beberapa dapat dikatakan sebagai *best practices* dalam perancangan sistem.

Istilah *design patterns* dimulai di bidang perancangan bangunan oleh Christopher Alexander. Dalam bukunya *A Pattern Language*, ia menerangkan pola-pola yang terdapat di dalam berbagai rancangan arsitektur bangunan. Arti *design pattern*diterangkannya dalam kalimat berikut:

"Each pattern describes a problem which occurs over and over again in our environment,and then describes the core of the solution to that problem, insuch a way that you can use this solution a million times over, without everdoing it the same way twice.".

Pada Gang of Four Patterns, katalog design patterns bisa dikategorikan menjadi tiga yaitu c*reational*, *structural*, dan *behavioral*. *Creational patterns* berhubungan dengan penciptaan obyek. Pola-pola ini berkisar seputar obyek mana yang diciptakan, siapa yang menciptakannya, serta berapa banyak obyek diciptakan. *Structural patterns* berhubungan dengan struktur statis obyek dan kelas. Pola-pola dalam *structural patterns* dapat dilihat pada saat program di-compile melalui struktur *inheritance*, properties, serta agregasi obyek-obyek. *Behavioral patterns* lebih berkenaan terhadap perilaku *run-time* program. Pola-pola ini berkaitan dengan algoritma serta interaksi antar obyek saat program berjalan.Penekanan *behavioral patterns* lebih pada komposisi obyek ketimbang *inheritance*. Secara umum membahasan tentang design pattern dapat

dijelaskan dengan bagaimana suatu design pattern akan digunakan yaitu:

- ✓ **Name** yaitu merupakan nama yang diberikan pada pola ini.
- ✓ **Intent** yaitu merupakan pernyataan ringkas yang memberikan permasalahan yang terjadi serta maksud yang hendak dicapai.
- ✓ **Also Known As,** Berbagai alias untuk pola ini, jika ada.
- ✓ **Motivation,** Sebuah skenario yang menerangkan sebuah permasalahan rancangan dan bagaimana pola ini dapat memecahkannya.
- ✓ **Applicability** yaitu Berbagai situasi di mana pola ini dapat diterapkan.
- ✓ **Structure** yaitu Sebuah gambar yang menerangkan hubungan kelas dan obyek dalam pattern ini.
- ✓ **Participants** yaitu Berbagai kelas dan/atau obyek yang turut serta dalam pola ini beserta peranannya.
- ✓ **Collaborations** yaitu Bagaimana kerja sama dari para peserta untuk melaksanakan peranannya masing-masing.
- ✓ **Consequences** yaitu Bagaimana pola ini mencapai tujuannya serta kompromi (*tradeoff*) yang harus dilakukan dalam penerapannya.
- ✓ **Implementation** yaitu Petunjuk, peringatan, serta berbagai teknik yang digunakan dalam penerapan pola ini.
- ✓ **Sample Code** yaitu Contoh program yang mengilustrasikan penerapan pola ini.
- ✓ **Known Uses** –yaitu Contoh-contoh dari penggunaan pola ini pada sebuah system sungguhan.
- ✓ **Related Patterns** yaitu Pola-pola lain yang berhubungan dengan pola ini.

Berikut ini adalah catalog design patterns menurut Gang of Four Patterns, dimana dibedakan berdasarkan fungsi dan kegunaannya:

Creational Patterns

Creational patterns berhubungan dengan penciptaan obyek. Pola-pola ini berkisar seputar obyek mana yang diciptakan, siapa yang menciptakannya, serta berapa banyak obyek diciptakan.

Tabel 1. Creational Pattern

Nama Pattern	**Pencipataan**	**Kesatuan**	**Tindakan**	**Tekanan**
Abstract Factory	Menciptakan	Sekumpulan obyek yang berhubungan atau saling tergantung	dengan menyediakan suatu antarmuka untuk melakukannya	Tanpa menentukan suatu kelas yang nyata.
Factory Method	Menciptakan		dengan menyediakan suatu antarmuka yang memungkinkan subkelas-subkelas untuk memutuskan obyek mana yang akan diciptakan.	Factory method memungkinkan sebuah kelas untuk menunda pembuatan keberadaan obyek kepada subkelas-subkelas.
Singleton	Jangan Menciptakan	lebih dari satu keberadaan dari suatu kelas	dengan menyediakan suatu titik akes global kepadanya.	

Structural Patterns

Masing-masing pola dalam *structural patterns* diuraikan menjadi empat kolom:

- ✓ **Tindakan** untuk diterapkan – apa yang dilakukan oleh pola yang bersangkutan.
- ✓ **Kesatuan** – perihal yang terkait dengan pola ini.
- ✓ **Akibat** – Dampak yang terjadi dari penerapan pola ini.
- ✓ **Hasil** – Apa saja yang didapatkan dari penerapan pola ini.

Tabel 2. Structural Patterns

Nama Pattern	Tindakan untuk diterapkan	Kesatuan	Akibat	Hasil
Bridge	Memisahkan satu dari yang lainnya	sebuah abstraksi dan penerapanny a		sehingga keduanya dapat berbeda dan tidak saling tergantung. .
Composite	Menyusun	obyek-obyek	ke dalam struktur pohon untuk mewakili hirarki seluruh-sebagian.	Composite memungkinkan klien klien untuk memperlakukan obyek tunggal maupun komposisi dengan cara yang sama.
Decorator	Secara dinamis memberikan Tambahan tanggungjawab kepada	sebuah obyek.		Decorator memberikan mekanisme perluasan yang mudah disesuaikan.
Façade	Memberikan sebuah kesatuan antarmuka kepada	suatu himpunan antar muka antar muka di dalam sebuah		Façade menetapkan antarmuka dengan tingkat yang lebih tinggi yang

		subsistem.		mempermudah penggunaan sebuah subsistem.
Flyweight	Penggunaan secara bersamasama dari	obyek-obyek		Flyweight memungkinkan sejumlah besar dari obyek-obyek yang berukuran kecil untuk ditangani secara tepatguna.

Behavioral Patterns

Masing-masing pola dalam *behavioral patterns* diuraikan menjadi tiga kolom:

- ✓ **Tujuan** – Hal yang ingin dicapai dari penerapan pola.
- ✓ **Rangkaian tindakan** – cara pola ini diterapkan.
- ✓ **Hasil** – Apa saja yang didapatkan dari penerapan pola ini.

Tabel 3. Behavioral Patterns

Nama Pattern	**Tujuan**	**Rangkaian tindakan**	**Hasil**
Command	Memparameterkan klien dengan permintaan-permintaan yang berbeda	dengan cara membungkus sebuah permintaan di dalam sebuah obyek.	Command memungkinkan Anda untuk mengantrikan atau mencatat permintaanpermintaan dan mendukung operasi yang dapat dibalikkan.
Iterator	Mengakses unsur-unsur dari		Iterator memberikan suatu jalan untuk

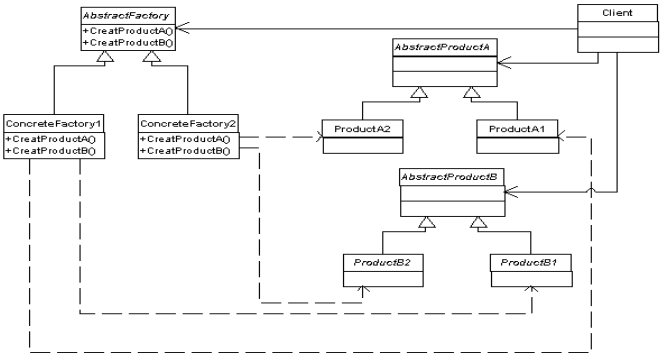
	sekumpulan obyek secara berurutan.		mengakses unsur-unsur tanpa menyingkap implementasi yang mendasarinya.
Observer	Semua obyek yang ergantung diberitahukan serta diperbaharui ketika suatu obyek berganti keadaan	dengan menetapkan hubungan satuke-banyak antara obyek-obyek.	.
Memento	Menangkap dan engeluarkan keadaan dalam suatu obyek sehingga keadaan ini dapat dikembalikan belakangan.		Memento melakukan hal ini tanpa melanggar pembungkusan.
Strategy	Membuat algoritma-algoritma dari suatu keluarga algoritma dapat dipertukarkan	dengan cara membungkus masing-masing algoritma.	Strategy memungkinkan algoritma berubah-ubah terlepas dari klien-klien yang menggunakannya.
Template Method	Menunda langkah-langkah suatu algoritma ke subkelassubkelas	dengan cara menetapkan rangka dari suatu algoritma di satu operasi.	Template Method emungkinkan subkelas-subkelas untuk menetapkan ulang langkah-langkah tertentu dari suatu algoritma tanpa mengubah susunan algoritma tersebut.

Pembahasan

Berikut ini adalah beberapa contoh pembahasan dari beberapa design pattern disesuaikan dengan kasus yang ditanganinya:

Abstract Factory Pattern

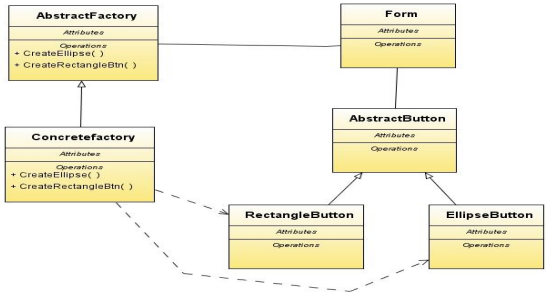
Tujuan design pattern ini adalah menyediakan interface dalam rangka menciptakan dependent object tanpa menetapkan konkret class-nya. Strucktur Abstract Factory Pattern adalah :



Gambar 1 Strucktur Abstract Factory Pattern

Contoh Kasus: Menciptakan beberapa jenis button yang memiliki tampilan yang berbeda.

Bentuk Class UML:



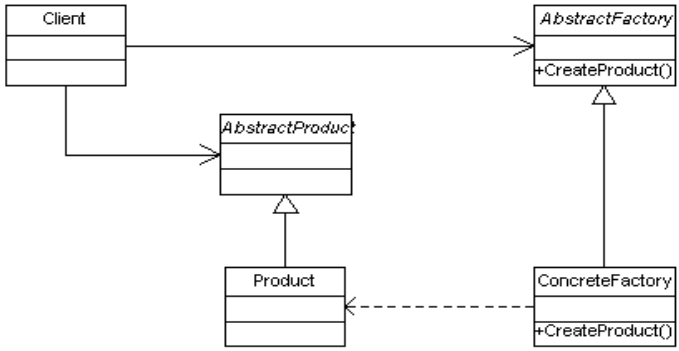
Gambar 2 Bentuk class UML Abstract Factory Pattern

“Membuat Sebuah Class Abstarct factory yang berfungsi sebagai interface untuk menciptakan masing-masing button dan class Concretefactory sebagai concrete subclass yang menciptakan button”

Factory Method Pattern

Tujuan pattern ini adalah mendefinisikan interface untuk menciptakan suatu objek dengan membiarkan subclass-nya memutuskan class mana yang direpresentasikan dengan contoh konkret.

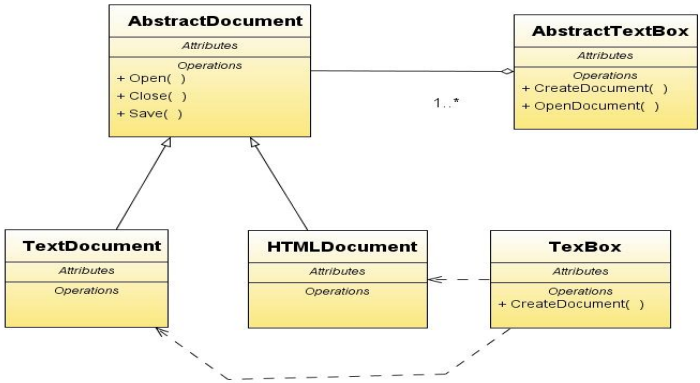
Structur Factory Method Patterns:



Gambar 3 Struktur Factory Method Pattern

Contoh kasus: bagai mana sebuah aplikasi dapat menyajikan berbagai dokumen dalam sebuah TextBox.

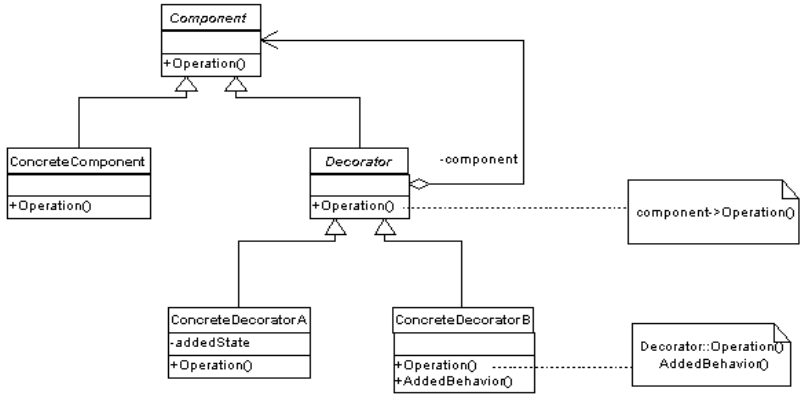
Bentuk Class UML:



Gambar 4 Bentuk class UML Struktur Factory Method Pattern

Decorator Pattern

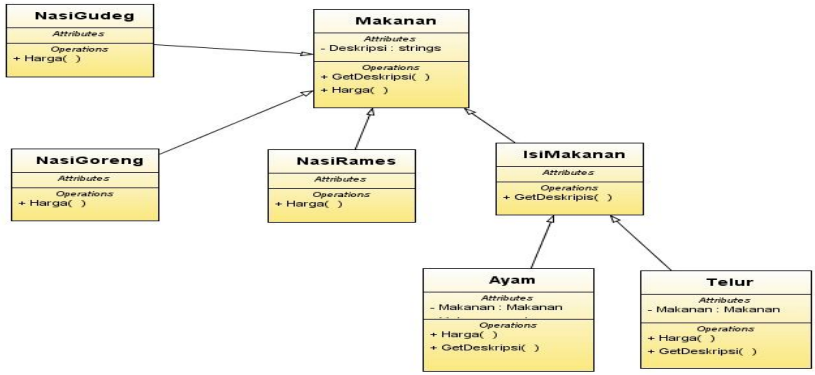
Tujuan Pattern ini adalah menciptakan suatu mata rantai object, dimana setiap object mempunyai objects responsible pada fungsi objek berikutnya.(The *Decorator* attaches additional responsibilities to an object dynamically.). Struktur Decorator Pattern:



Gambar 5 Struktur decorator Pattern

Contoh kasus: Memesan nasi dengan berbagai jenis nasi yang dipilih berdasarkan menu.sebagai contoh akan memesan nasi yaitu :Memesan nasi gudeg → Menambahkan daging ayam suwir→Menambahkan sepotong telur.

Bentuk Class UML:

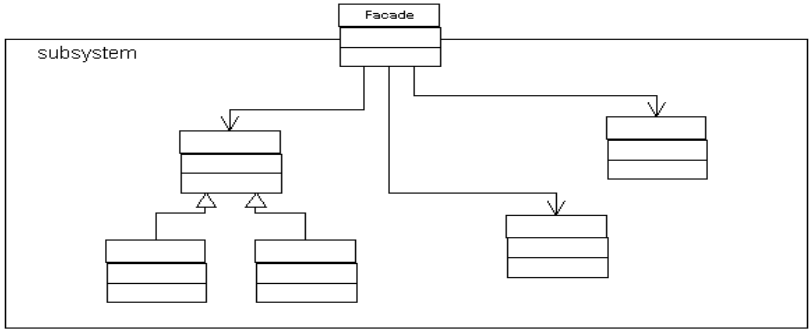


Gambar 6 Bentuk class UML decorator Pattern

Façade Pattern

Tujuan adalah untuk menyediakan unifikasi interface pada sekumpulan interface yang terdapat pada sebuah subsystem, dimana façade akan mendefiniskan interface tingkatan yang lebih tinggi sehingga subsystem lebih muda digunakan dengan menguraikan system kedalam subsystem untuk mengatasi kompleksitas sebuah system.

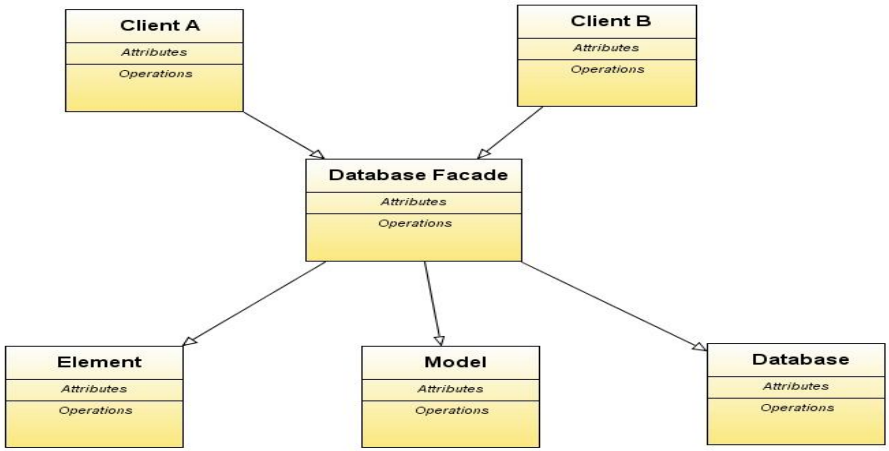
Structure Façade Pattern:



Gambar 7 Struktur Facade Pattern

Contoh Kasus: Bagaimana ada suatu client yang berkomunikasi dengan suatu database, model dan element, dimana client harus membuka suatu database untuk mendapatkan model dan melakukan query model untuk mendapatkan element.

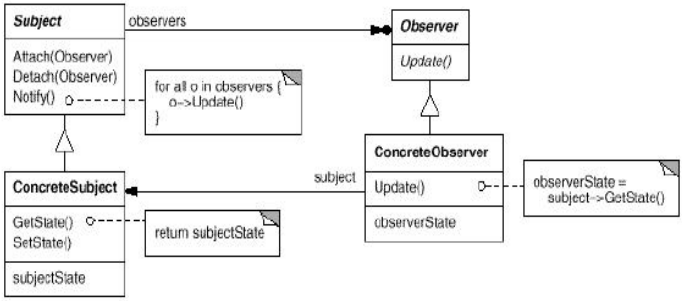
Bentuk Class UML :



Gambar 8 Bentuk Class UML Facade Pattern

Observer Pattern

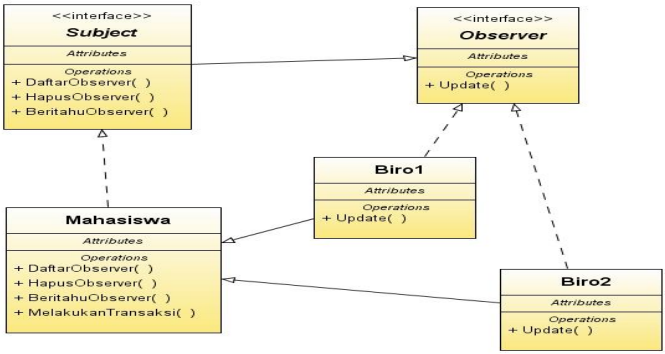
Tujuan pattern ini adalah mendefiniskan hubungan one to many antar objek sehingga ketika sebuah objek berubah state-nya, objek-objek lain yang tergantung atau berhubungan dengan objek tersebut akan diberitahu dan diupdate secara langsung.
Struktur Observer Pattern:



Gambar 9 Struktur Observer Pattern

Contoh kasus: Menentukan status registrasi mahasiswa oleh suatu biro pengecekan. Sebagai contoh ada biro 1 dan biro 2 yang akan melakukan pengecekan status dari registrasi mahasiswa.

Bentuk Class UML:



Gambar 10 Bentuk Class UML Observer Pattern

Penutup

Dalam membangun suatu perangkat lunak yang berbasiskan objek diperlukan suatu aturan atau pola untuk mendesain sebagai keperluan dokumentasi dan design patterns bukan sekedar menyediakan solusi terbaik dalam memecahkan suatu masalah, tetapi lebih dari pada itu patterns membuat design software menjadi lebih efektif, fleksible, resilient dan waktu penyelesaikan desain software juga lebih efisien.

Daftar Pustaka

Alexander, Christopher (1977): A pattern language, Oxford University Press

Ridi, Modul Kuliah Design Patterns, MTI UGM Yogyakarta, 2007

http://www.st.cs.uni-sb.de/~brudaru/misc/pattern-examples.pdf, diakses 11 januari 2008

http://www.daniellafreniere.com/PDF/Pattern_Tutorial.pdf, diakses 11 Januari 2008

http://www.ibm.com/developerworks/rational/products/patternsolutions/, diakses Desember 2007