BAB V
Abstraksi dan Generalisasi

Abstraksi

Abstraksi adalah deskripsi dari suatu masalah pada level generalisasi tertentu, sehingga memungkinkan kita untuk berkonsentrasi pada aspek kunci dari masalah tersebut tanpa memperhatikan hal-hal detail.

Abstraksi dapat membantu kita untuk fokus pada hal-hal penting dari suatu masalah.

Abstraksi melibatkan pengidentifikasian kelas-kelas (classes) dari suatu object, sehingga memungkinkan kita menggroupkannya. Dengan cara tersebut kita bekerja dengan sedikit parameter/variabel dari kelas-kelas yang ditinjau.

Contoh :
- Monitoring : berbagai macam sistem monitoring
- Ban sepeda : sepeda balap, sepeda gunung
- Mobil : sedan, jeep, wagon, truk, dll

Tingkatan Abstraksi

- Abstraksi Fungsional
  Komponen mengimplementasikan satu fungsi, misalnya fungsi matematika. Pada intinya interface merupakan fungsi itu sendiri.

- Pengelompokkan Kasual
  Komponen merupakan sekumpulan entitas yang berhubungan longgar (loosely related) yang mungkin berupa deklarasi data, fungsi, dsb. Interface terdiri dari nama semua entitas pada pengelompokan tersebut.

- Abstraksi Data
  Komponen merepresentasikan abstraksi data atau kelas perangkat lunak bahasa berorientasi obyek. Interface terdiri dari operasi untuk membuat, memodifikasi dan mengakses abstraksi data.

- Abstraksi Cluster
  Komponen merupakan sekumpulan kelas yang berhubungan yang bekerja sama. Kelas-kelas ini kadang-kadang dinamakan kerangka kerja. Interface merupakan komposisi semua interface dari obyek-obyek yang membangun kerangka kerja tersebut.

- Abstraksi System
  Komponen merupakan system yang sepenuhnya berdiri sendiri. Pemakaian ulang abstraksi tingkat system kadangkala disebut pemakaian ulang produk cost. Interface adalah apa yang disebut API (Aplication Programming Interface) yang didefinisikan untuk memungkinkan program mengakses command dan operasi.

Generalisasi

Generalisasi adalah perluasan suatu aplikasi yang meliputi suatu daerah obyek yang lebih besar dengan jenis yang berbeda atau jenis yang sama.

Binding

- Attribute : nilai internal atau data terkait pada suatu obyek yang menunjukkan ciri-ciri atau sifat-sifat dari obyek serta penggambaran keadaan (state) obyek
  Contoh :
  Nama obyek : mobil
  Attribute :
  Merek : Toyota
  Silinder : 2000 cc
  Warna : merah
  Status : baru / jalan
  Tahun : 2006

- Binding : Pengaturan nilai attribute
- Descriptor : informasi attribute yang diisikan dalam tempat penyimpanan untuk setiap entitas

Binding merupakan pusat dari konsep definisi semantik bahasa pemrograman.

Bahasa pemrograman berbeda satu dengan yang lainnya karena :
a. perbedaan jumlah entitas yang dapat ditangani
b. jumlah attribute yang dapat ditempelkan ke entitas yang dapat ditangani
c. waktu kemunculan binding (binding time)
d. stabilitas binding (binding yang sudah terbentuk bersifat tetap atau dapat dimodifikasi)

Jenis Binding

Ada dua tipe / jenis binding bila dilihat dari kontrol yang digunakan untuk binding data, yaitu :
a. Simple Binding
b. Complex Binding

Jika dilihat dari sisi waktu pengikatan data (binding) dapat dibedakan lagi ke dalam dua jenis, yaitu :

a. Early Binding
b. Late Binding

Contoh Binding

- Language definition time binding
  Dalam banyak bahasa pemrograman (Fortran, ADA, C++), tipe "integer" di-binding-kan pada waktu pendeteksian bahasa untuk operasi matematika yang sudah umum, misalnya untuk operasi aljabar yang menghasilkan dan memanipulasi "integer".

- Language implementation time binding

Dalam banyak bahasa pemrograman (Fortran, ADA, C++) suatu kumpulan nilai di-binding ke tipe integer pada waktu implementasi bahasa. Pada saat pendefinisian bahasa, tipe integer harus didukung dan implementasi bahasa mem-binding-kan ke representasi memori, yang kemudian menentukan sekumpulan nilai yang akan diisikan ke dalam tipe tersebut.

- Compile-time (Translation-time) binding
  Pascal mempunyai fasilitas definisi predefine dari tipe "integer", tetapi mengizinkan programmer untuk mendefinisikan ulang tipe ini. Dengan demikian, tipe "integer" di-binding-kan sebagai gambaran pada waktu impementasi bahasa, tetapi binding ini dapat dimodifikasi pada saat translation time.

- Execution-time (Run-time) binding
  Pada banyak bahasa pemrograman, variabel di-binding-kan ke suatu nilai pada saat execution time, dan binding dapat dimodifikasi berulang-ulang selama eksekusi.

Enkapsulasi

- Pengkapsulan berarti mengemas beberapa item bersama-sama menjadi satu unit yang tertutup dalam rangka menyembunyikan struktur internal suatu obyek dari lingkungan/dunia luar
- Pengkapsulan sering dianggap sebagai "penyembunyian informasi"
- Setiap kelas hanya menampakkan interface yang diperlukan untuk berkomunikasi dengan dunia luar melalui message dan menyembunyikan (encapsulating)/implementasi aktual di dalam kelas.
- Kita hanya membutuhkan pemahaman tentang interface (method), tidak perlu paham tentang internalnya(implementation)
- Pengkapsulan merupakan kemampuan sebuah obyek kelas untuk membatasi akses client ke representasi internal obyek (data dan fungsi)

Prinsip Generalisasi

Prinsip generalisasi adalah suatu bentuk umum dari suatu kesatuan yang khusus.
Contoh :

Lamda p.B'

Dimana lambda menyatakan suatu abstrak yang menandakan generalisasi B jika p dipanggil oleh suatu parameter B'

Prinsip generalisasi tergantung pada prinsip analogi.

Generalisasi dan abstrak sering digunakan bersam-sama. Abstrak digeneralisasi dengan parameterisasi untuk mendapatkan manfaat yang lebih besar. Di dalam parameterisasi atau lebih bagian dari satu kesatuan dapat digantikan dengan suatu nama baru.

Nama yang digunakan sebagai suatu paramenter ketika abstrak yang telah diparamenterkan dilibatkan dengan suatu binding parameter disebt argumentasi.

Prinsip Analogi

Prinsip analogi ada ketika suatu penyelesaian pada pola diantara 2 obyek yang berbeda. Dimana obyek dapat digantikan dengan obyek tunggal yang parameterized untuk melakukan rekonstruksi yang menyangkut obyek yang asli.

Prinsip Parameterisasi

Prinsip parameterisasi adalah suatu parameter yang secara umum mungkin berasal dari beberapa domain. Istilah parameter formal dan parameter nyata sering disebut dengan argumentasi.

Substitusi

Kegunaan abstraksi dan generalisasi tergantung pada substitusi

Prinsip Korespondensi

Prinsip korespondensi adalah suatu formalitas yang menyangkut aspek/pengarahan prinsip abstrak yang mengandung substitusi dan definisi yang saling terkait.

Struktur Block

- Block merupakan suatu bagian dari scope-defining bahasa pemrograman. Artinya, Block merupakan suatu definisi wilayah bagian bahasa pemrograman
- Block merupakan urutan dari statement yang executable yang diperlakukan sebagai suatu unit

Block disebut subprogram atau routine dikebanyakan bahasa pemrograman.

Struktur Block dari suatu bahasa pemrograman :

```
Program main;
--deklarasi lokal Main;
Procedure Subpro1;
---deklarasi lokal Subpro1;
       Procedure Subpro3;
       ---deklarasi lokal Subpro3;
Procedure Subpro4;
---deklarasi lokal Subpro4;
Begin
--Statement untuk Subprog1;
End Subprog1;
Procedure Subprog2;
--Deklarasi lokal Subprog2;
Begin
--statemen untuk Subpro2;
End Subprog2;
Begin
--statemen untuk main;
End main
```

Aturan Cakupan :

a. Dynamic Scope

Suatu subprogram yang didefinisikan di satu tempat dalam suatu program dan dapat dipanggil dari lingkungan yang berbeda (lingkungan dimana subprogram tersebut tidak didefinisikan)

b. Static Scope
Subprogram dipanggil dari lingkungan tempat subprogram tersebut didefinisikan

Lingkungan

Lingkungan lokal suatu subprogram Q terdiri atas bermacam-macam identifier yang dideklarasikan di bagian atas dari subprogram Q.

Nama, variabel, nama parameter formal, dan nama subprogram diperhatikan disini. Nama subprogram disini merupakan nama subprogram yang didefinisikan secara lokal di dalam subprogram Q (nested subprogram).