PENGGUNAAN METODE PEMROGRAMAN BERORIENTASI OBJEK (*OBJECT ORIENTED PROGRAMMING*) PADA BAHASA PEMROGRAMAN

Oleh : Priatno

ABSTRACT

The bigger a system applied in a company or an institution the greater the problems it causes. Such condition requires an establishment and development of computer based programming as one alternative solution to the problems.

Object Oriented Programming is a programming application, which is widely used among programmers when they build a programming application to solve problems faced by a computerized system.

Semakin besar suatu system pada perusahaan atau lembaga otomatis akan menambah juga permasalahan yang dapat ditimbulkan sebagai akibat dari hal tersebut dan hal ini membutuhkan pembangunan dan pengembangan secara significan aplikasi pemrograman berbasis computer sebagai salah satu alternative pemecahan dari permasalahan yang timbul

Pemrograman beroientasi objek (Object oriented programming) saat ini merupakan bentuk aplikasi pemrograman yang sangat banyak digunakan dan sangat disukai dikalangan programmer dalam membuat suatu program aplikasi untuk menyelesaikan suatu permasalahan dari suatu system yang menggunakan komputer sebagai alat bantunya.

I. PENDAHULUAN

Metode ini adalah meningkatkan produktivitas pemrograman dengan meningkatkann extensibilitas dan penggunaan kembali perangkat lunak dan mengontrol biaya dan kompleksitas perawatan perangkat lunak.

Akan mampu melaksanakan pembelajaran pemrograman orientasi objek dengan menerapkan pengetahuan dan keterampilan tentang pemrograman yang relatif merupakan metode baruuntuk melakukan perancangan atau rekayasa perangkat lunak.

II. PEMBAHASAN

Pemrograman berorientasi objek merupakan objek yang relative baru untuk melakukan perancangan atau rekayasa perangkat lunak.

Pemrograman berorientasi objek bukan hanya merupakan teknik penulisan program, tetapi lebih merupakan teknik pemaketan program yang memungkinkan produsen program mengkapsulkan fungsi-fungsi yang ada dalam suatu program, yang ditujukan untuk konsumen program merupakan batas yang memisahkan teknik pemrograman konvensional dengan pemrograman berorientasi objek. Alat Bantu pemrograman konvensional menekankan hubungan pemrograman dengan programnya. Teknik pemrograman berorientasi objek menekankan hubungan antara produsen dengan konsumen program. Dalam pemrograman berorientasi

objek, bentuk-bentuk objek bervariasi tergantung pada persoalannya. Objek dapat berupa angka, rangkaian karakter, antrian, kamus, program, compiler, proses komputasi, riwayat keuangan, dan sebagainya. Objekdidefinisikan sebagai beberapa datum privat dan sekumpulan prosedur atau metode yang dapat mengakses data tersebut. Jenis operasi terhadap objek tergantung pada jenis komponen yang direpresentasikan oleh objek tersebut. Arsitektur pemrograman berorientasi objek dibangun di lingkungan kelas-kelas yang akan mendasari semua sifat kegiatan dalam teknik pemrograman tersebut. Objek-objek dari kelompok kelas tertentu dimanipulasi dengan memanggilan metode di dalam kelas tersebut. Contohnya, mengirim pesan kepada objek. Pesan mewakili aksi-aksi yang akan dikenakan terhadap objek. Pemrograman berorientasi objek memusatkan perhatian pada prosedur yang akan melakukan manipulasi sehingga data membentuk basis dekomposisi perangkat lunak.

Dalam istilah pemrograman berorientasi objek, interface ke metode yang didefinisikan dalam suatu kelas dapat dipisahkan drai detil pada saat implementasi program. Pemisah konsep ini merupakan hal yang mendasar yang sangat penting dalam pemrograman berorientasi objek, selain penggunaan kembali program yang ada dan kendali atas biaya perawatan perangkat lunak tersebut.

Penggunaan kembali program semakin dimungkinkan dalam pemrograman berorientasi objek karena konsep yang dikapsulkan dalam bentuk kelas sudah tersedia dalam interface pesan. Pemakai hanya perlu memahami sifat-sifat setiap interface pesan, tanpa harus mengetahui secara detil tentang implementasi suatu program computer. Secara singkat, tujuan utama pengembangan perangkat lunak atau program berorientasi objek adalah sebagai berikut :

- Mempersingkat waktu dan menurunkan biaya pengembangan rekayasa perangkat lunak.
- Menurunkan biaya perawatan perangkat lunak.
- Pemrograman berorientasi objek memberi landasan yang sangat berguna untuk pembuatan prototype sistem secara cepat.

Metode dan Objek Yang Digunakan.

Metode merupakan suatu prosedur atau fungsi yang disatukan dalam suatu objek dan dapat mengakses field-field data yang menjadi bagian dari objek tersebut. Berikut salah satu contoh program pemberian nilai ke dalam data yang bertype record.

```
Type lokasi = record
Kolom, baris : BYTE;
END;
PROCEDURE intilokasi (VAR Target : Lokasi;
LokX, LokY : BYTE);
BEGIN
Target.Kolom :=LokX
Target.Baris :=LikY
END;
Var Lokasinya ; Lokasi ;
BEGIN
Intilokasi(Lokasinya, 10, 10);
END.
```

Perhatikan program sederhana di atas, pada Intilokasi dilakukan pemberian nilai dari LOkX an LokY ke target yang bertipe record. Mungkin pertanyaan muncul, mengapa pada prosedur Intilokasi harus dikhususkan

tipe recordnya? Ada cara untuk menggabungkan tipe record dan prosedur untuk melakukan inisialisasi tersebut, yaitu menggunakan metode (dalam suatu objek). Perhatikan program berikut:

```
TYPE Lokasi = OBJECT
        Kolom, Baris, : BYTE ;
        PROCEDURE Intilokasi (Klm, BRS
: BYTE);
END;
PROCEDURE Lokasi.Intilokasi (Klm, Brs
:BYTE);
BEGIN
        Kolom := Klm;
        Baris := Brs;
END;
Var Lokasinya : Lokasi;
BEGIN
        Lokasinya.Intilokasi ( 10, 10);
END.
```

Di dalam sebuah objek metode didefinisikan sebagai sesuatu header fungsi atau prosedur yang bertindak sabagai metode dan semua field data harus didefinisikan sebelum deklarasi metode pertama. Salah satu contonya sebagai berikut :

```
TYPE Lokasi = OBJECT
        Kolom, Baris : BYTE;
        PROCEDURE Intilpkasi (Klm, Brs :
BYTE);
END;
```

Di sisi lain di luar objek, sebuah metode didefinisikan secara penuh, yaitu statemen-statemen apa saja yang akan dikerjakan oleh metode tersebut. Nama metode harus diletakkan setelah nama objek yang memiliki metode tersebut.

Contohnya :

```
PROCEDURE Lokasi.Intilokasi (Klm,Brs :
BYTE);
BEGIN
        Kolom := Klm;
        Baris := Brs;
END;
```

Di dalam pemrograman berorientasi objek, sebuah objek sama seperti sebuah record yang terdiri darai beberapa field data dengan diperluas oleh adanya fungsi atau prosedur.Bentuk umum pendifinisian sebuah objek adalah sebagai berikut :

```
Type
ObjekBaru =OBJECT [(objek induk)
Fielddata;
        .....
Field data;
metode;
        .....
        Metode;
END;
```

Bandingkan bentuk umum tersebut dengan bentuk umum pendefinisian sebuah record.

```
Type
Recordbaru = RECORD
        field data;
        .....
        field data;
END;
```

Satu contoh lagi pendefinisian sebuah objek titik pada suatu lokasi dilayar monitor sebagai berikut :

```
TYPE Lokasi = OBJECT
        Kolom, Baris : BYTE;
END:
Titik = OBJECT (Lokasi)
        Menyala : BOOLEAN
        PROCEDURE Initial (Klm, Brs :
BYTE);
        PROCEDURE Tampak
        PROCEDURE Hilang
END:
```

Dikarenakan tipe data recordnya hanya berisi data field saja, jika objek diatas didefinisikan sabagai record, maka :

```
TYPE Lokasi = RECORD
        Kolom, Baris ; BYTE;
END;
Titik = RECORD
```

```
        Posisi : Lokasi;
        Menyala : BOOLEAN;
End
```

Mengakses field-field atau metode-metode suatu objek dapat kita lakukan seperti mengakses field-field record,yaitu dapat menggunakan titik atau statemen WITH.
Dot merupakan suatu contoh (*instance* atau variable yang bertiope objek) dari objek titik.

```
Var Dot : Titik;
BEGIN
        Dot. Menyala := TRUE; (mengakses field objek secara langsung)
        Dot. Inisial (10, 10)
        Dot. Hilang;
END
```

Hal diatas dapat dilakukan dengan menggunakan statemen WITH, WITH Dot DO BEGIN

```
        Menyala := TRUE; (mengakses filed objek secara langsung )
        Inisial (10, 10);
        Tampak;
        Hilang;
END;
```

Walaupun kita dapat mengakses field-field objek secara langsung, namun hal tersebut bukan suatu hal yang baik. Prinsip pemrograman berorientasi objek adalah sedapat mungkin field-field dalam suatu objek ditinggalkan menyendiri sebanyak mungkin, jangan sampai field-field data tersebut dapat diakses secara langsung. Lalu bagaimana cara mengakses field-field data tersebut? Jawabannya adalah sedapat mungkin menggunakan metode objek (yaitu prosedur atau fungsi yang didefinisikan di dalam objek) yang bersangkutan.

Ada 3 karakteristik utama bahasa pemrograman berorientasi objek yaitu :

a. Enkapsulasi (pengkapsulan)

Enkapsulasi merupakan gabungan beberapa type data (record) dengan prosedur dan fungsi yang membentuk suatu type data baru yang tepat. Hakikat dari enkapsulasi ini adalah memadukan langkah program dengan data yang ada di dalamnya. Dengan enkapsulasi anda dapat menyembunyikan di dalam objek, baik data maupun fungsi / prosedur. Hanya dengan interface objek maka kita dapat mengakses datanya.
Perhatikan 2 buah contoh di bawah ini :

```
PROGRAM Mobil;
User Crt;
CONST Rumah=10;
                        Tujuan:INTEGER=R
umah;
VAR Posisi : INTEGER;
                    Maju : BOOLEAN;
{pesen mobil mulai berangkat
PROCEDURE mobil mulai jalan;
BEGIN
                        WRITELN ('mobil
mulai berjalan.....');READLN;
END;
{majukan satu langkah posisi mobil, dan cek apakah sampai tujuan
FUNCTION Sampai_Tujuan : BOOLEAN;
BEGIN
    WRITELN ('sedang mengendarai mobil...');READLN
    INC (posisi);
    IF (posisi = Tujuan) THEN
        Sampai_Tujuan := TRUE
ELSE
        Sampai_Tujuan :=FALSE
END;
{pesan saat mobil direm
PROSEDURE Rem Mobil;
BEGIN
    WRITELN ('Mobil direm....');
READLN;
```

```
END;
{jika ditemui rintangan, mobil harus
mundur
PROCEDURE Mobil Mundur;
BEGIN
    IF (Maju)THEN BEGIN
        WRITELN ('Mobil
Mundur.....');READLN
        DEC (Posisi);
        Maju :=FALSE
    END
    ELSE Maju :=TRUE;
END;
{pesan mesin mobil dimatikan
PROCEDURE Matikan_Mesin;
BEGIN
        WRITELN('Matikan mesin
mobil');READLN;
END;
{cek apakah ada rintangan dijalan
FUNCTION Ada_Rintangan : BOOLEAN;
BEGIN
        IF RANDOM(4) > 0 THEN
                Ada_Rintangan :=TRUE
        ELSE
                Ada_Rintangan :=
FALSE
END;
VAR
        Di_tujuan, {apakah sampai
tujuan
        Rintangan : BOOLEAN;
{apakah ada rintangan
BEGIN
        RANDOMIZE;
        CLRSCR;
                Mobil_Mulai Jalan;
    Di tujuan := FALSE;
                Maju :=TRUE;
    Posisi :=0
    WHILE NOT (di_tujuan) DO BAGIN
    Di_tujusn :=Sampai_Tujuan;
    Rintangan :=Ada_Rintangan;
    IF (Rintangan AND NOT (di_tujuan))
THEN BEGIN
        WRITELN ('Awas! Ada sesuatu
di jalan....');
        READLN;
        Rem_mobil;
        Mobil_mundur;
        Mobil_mundur;
    END;
                END;
                WRITELN (Nah, telah
sampai di rumah...');
                Matikan_=Mesin
END.
```

Program di atas menggunkan beberapa variable global, yaitu posisi, maju, dan tujuan. Statemen RAMDOMIZE digunakan untuk menghasilkan nilai ramdom untuk menstimulasi ada atau tidaknya rintangan sebuah looping WHILE selama nilai di_tujuan adalah FALSE. Dalam looping, mobil akan terus maju (yaitu dengan sampai_tujuan) dan jika nilai posisi = tujuan, maka nilai di_tujuan adalah TRUE.

Selama perjalanan, mungkin ditemui rintangan yang ditentukan dengan fungsi Ada_Rintangan. Jika rintangan ditemukan dan belum sampai ditujuan, maka mobil harus direm dan mundur satu langkah (dengan mobil_mundur), kemudian maju kembali. Perhatika program kedua yang merupakan program OOP untuk simulasi mobil diatas sbagai berikut :

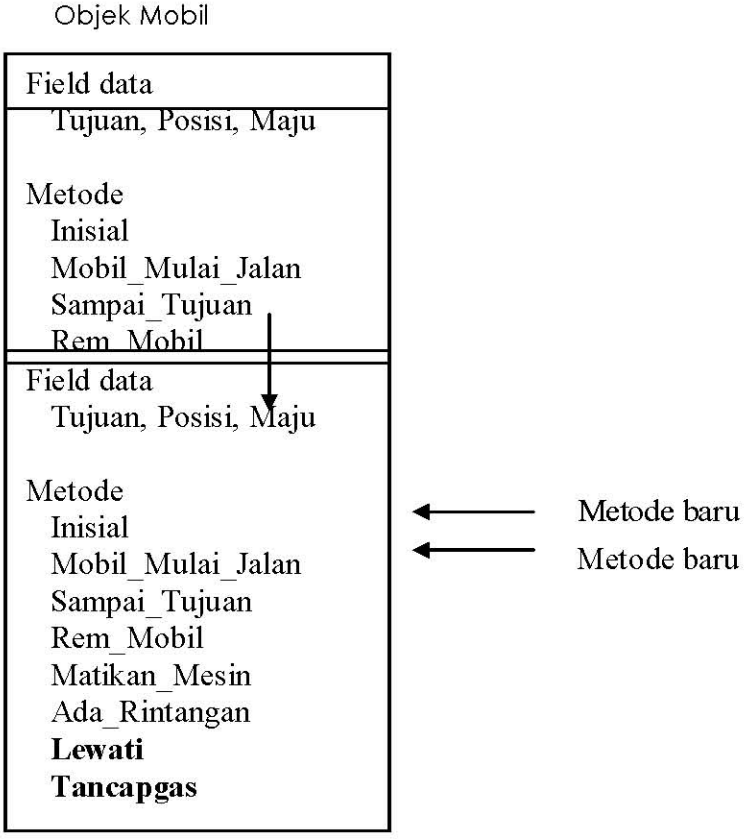
```
PROGRAM Mobil2;
USES Crt;
CONST Rumah = 10;
Type Mobil = OBJECT
        Tujuan, Posisi : INTEGER;
        Maju : BOOLEAN;
        PROCEDURE inisial;
        PROCEDURE
Mobil_Mulai_Jalan;
        PROCEDURE Sampai_Tujuan;
        PROCEDURE Rem_Mobil;
        PROCEDURE Mobil_Mundur
        PROCEDURE Matikan_Mesin
        PROCEDURE Ada_Rintangan ;
BOOLEAN;
END;
```

Pemberian nilai awal kondisi mobil sebelum berangkat

b. Inheritance (penurunan sifat)

Penurunan sifat memungkinkankita membuat sebuah objek baru yang sama dengan objek sebelumnya yang telah didefinisikan. Karateristik suatu objek diturunkan ke objek yang lain sehingga objek yang baru akan memiliki karakterstik yang sama dengan induknya. Penurunan ini di awali dengan mendefinisikan objek induk. Berdasarkan objek induk tersebut dapat dibuat objek keturunan. Setiap objek turunan dapat mengakses semua data dan program yang dimiliki objek induknya.

Kembali ke program simulasi mobil di atas, misalnya kita ingin membuat objek mobil yang memiliki kecepatan dua kali kecepatan mobil biasa. Kita dapat melakukan penurunan sifat objek mobil yang telah didefinisikan . selanjutnya, tambahkan karakteristik pada objek baru tersebut untuk menangani kacepatan mobil. Berikut programnya :




```
USES Crt;
CONST Rumah =  10;
TYPE Mobil = OBJEK
Tujuan, Posisi : INTEGER;
Maju           : BOOLEAN;
  PROCEDURE Inisial;
  PROCEDURE Mobil_Mulai_Jalan;
  PROCEDURE  Sampai_Tujuan      :
BOOLEAN ;
  PROCEDURE Rem_Mobil;
  PROCEDURE Mobil_Mundur;
  PROCEDURE Matikan_Mesin;
```

```
  FUNCTION  Ada_Rintangan  :
  BOOLEAN ;
END;
{ mobil cepat diturunkan dari obyek
mobil dengan ditambah
{ sebuah fungsi yang dapat melewati
rintangan dengan
{ kecepatan 2 x dari kecepatan mobil
induknya
MobilCepat = OBJECT ( Mobil )
   PROCEDURE Lewati;
   FUNCTION TancapGas : BOOLEAN
END;
……..
Berisi prosedur dan fungsi yang sama
dengan program 2
……..
Lewati rintangan dengan menambah
kecepatan
PROCEDURE MobilCepat . Lewati;
BEGIN
        Posisi := Posisi + 2;
END;
Tampilkan jika ada rintangan dan cek
posisinya
FUNCTION MobilCepat . TacapGas :
BOOLEAN;
BEGIN
    WRITELN ('Lewati rintangan.....');
    READLN;
    Lewati; memanggil fungsi
    MobilCepat . Lewati
    IF ( Posisi >= Tujuan ) THEN
    TancapGas := TRUE
    ELSE
    TancapGas := FALSE;
END;
VAR
    Di_tujuan,
    Rintangan : BOOLEAN;
    Mobilku : MobilCepat;
BEGIN
    RANDOMIZE;
    CLRSCR;
    Di_tujuan := FALSE;
Mobilku.Inisial;
Mobilku.Mobil_Mulai_Jalan;
WHILE NOT
Di_tujuan :=Mobilku.Sampai.Tujuan;
Rintangan :=Mobilku.Ada.Rintangan;
IF(Rintanagn ANDNOT(di_tujuan)) THEN
BEGIN
        WRITELN ('Awas! Ada sesuatu
dijalan....');
        READLN;
        Mobilku.Rem_Mobil;
        Mobilku.Mobil_Mundur;
        Mobilku.Mobil_Mundur;
        Di_tujuan
:=Mobilku.TancapGas;
END; END;
WRITELN ('Nah, telah sampai
dirumah....');
Mobilku.Matikan_Mesin;
END;
```

Pada contoh program diatas objek mobil cepat merupakan turunan dari objek yang secara tidak langsung semua field data dan metodenya akan dimiliki pula oleh objek mobil cepat. Untuk menunjukkan bahwa mobil cepat merupakan mobil yang berkecepatan lebih, perlu ditambahkan karateristiklain, dalam hal ini ditambahkan sebuah metode baru, yaitu lewati yang akan menambah 2 pada nilai popsisi jika mobil menemui rintangan di jalan dan tancapgas yang akan memanggil lewati dan mencek apakah sampai tujuan atau belum.

c. Polimorphism (keaneka ragaman)

Dengan polymorphism kita dapat membuat sebuah objek baru yang menunjukan fungsi-fungsi yang sama dengan objek induknya, namun fungsi-fungsi yang sama tersebut memiliki operasi yang berbeda-beda, objek yang masih di dalam suatu hierarki dapat menyusun objek yang lainnya, sesuai dengan kebutuhan dari objek tersebut, yang artinya pemrograman berorientasi objek ini mencakup keaneka ragaman objek yang menjamin terhimpunnya

perpaduan keaneka ragaman data dengan prosedur atau fungsi.

III. KESIMPULAN

Tujuan utama pengembangan perangkat lunak atau program berorientasi objek adalah sebagai berikut :

1. Mempersingkat waktu dan menurunkan biaya pengembangan rekayasa perangkat lunak.
2. Menurunkan biaya perawatan perangkat lunak.
3. Pemrograman berorientasi objek memberi landasan yang sangat berguna untuk pembuatan prototype sistem secara cepat.

3 Karakteristik pada ***Objek Oriented Programming***

1. Pengkapsulan (Encapsulation)
   a. Pengemasan Data Dan Prosedur Dlm Obyek.
   b. Upaya Penyembunyian Informasi Yg Diterapkan Pd Pmprograman Terstruktur Data Tak Dpt Diakses Scra Langsung Diluar Obyek.
2. PEWARISAN (INHERITANCE) Merupakan Sifat Pada Oop Yg Memungkinkan Sifat 2 Dari Kelas Diturunkan Kekelas Lain
3. Polimofisme
   a. Konsep Menyatakan Nama Yg Sama Dpt Memiliki Bentuk Dan Perilaku Yg Berbeda
   b. Terkait Dg Pewarisan,Jika Kelas Diturunka Dr Kelas Lain, Kelas Turunan Mewarisi Metode 2 Kelas Induknya (Dan Dapat Diubah)

DAFTAR PUSTAKA

Jogianto, HM.1993.*Bahasa Cobol.* Andi Offset, Yogyakarta

Kartono, Susanto. 1996. *Pemrograman berorientasi objek* dengan Borlan C++. Andi Offset, Yogyakarta.

Nugroho, Eko MS. 1996. *Pemrograman Terstruktur* dengan Pascal. AndiOffset, Yogyakarta