PENERAPAN *MULTY INHERITANCE* DALAM PARADIGMA BERORIENTASI OBJEK MENGGUNAKAN METODE COAD YOURDON

Iyat Ratna Komala

Dosen Jurusan Sistem Informasi STMIK Sumedang

Email : iratnak@stmik-sumedang.ac.id

ABSTRAK

Tujuan dari penelitian ini adalah untuk mengetahui dan membandingkan hasil penurunan sifat / data yang dimiliki oleh kelas anak dari kelas induk. Metode yang digunakan adalah dengan metode berorientasi objek dari Coad Yourdon dengan diimplementasikan menggunakan bahasa pemrograman C++. Hal ini didasarkan bahwa pada metode berorentasi objek memiliki kelebihan dibandingkan metode terstruktur. Lalu adanya konsep dan peran DNA dalam sebuah sel yang berfungsi sebagai materi genetik, maksudnya DNA akan menyimpan cetak biru bagi segala aktivitas sel, sehingga hal ini berkaitan dengan konsep objek dengan penurunan sifatnya yang dikenal dengan inheritance. Hasil yang diperoleh dari penelitian ini adalah bahwa kelas anak memang dapat mewarisi kelas induknya, dengan memiliki keterbatasan bahwa kelas anak yang diwarisi dari dua kelas induk dengan atribut yang sama maka akan menghasilkan sifat yang ambigu. Sehingga pada tahap identifikasi atribut setiap kelas idealnya tidak memiliki atribut yang berbeda yang akan diturunkan ke kelas yang sama. Contohnya ayah memiliki hidung mancung, ibu memiliki hidung pesek maka anak akan memiliki hidung,hal ini yang disebut ambigu.

Kata kunci : multy inheritance, paradigm metode berorientasi objek coad yourdon

PENDAHULUAN

Latar Belakang Masalah

DNA (bahasa Inggris: *deoxyribonucleic acid*), atau Asam deoksiribonukleat adalah sejenis asam nukleat yang tergolong biomolekul utama penyusun berat kering setiap organisme. Di dalam sel, DNA umumnya terletak di dalam inti sel. Peran DNA di dalam sebuah sel adalah sebagai materi genetik, artinya DNA menyimpan cetak biru bagi segala aktivitas sel. Ini berlaku umum bagi setiap organisme, termasuk hewan, binatang dan manusia. [6].Para ilmuwan menemukan pada "*Human Genome*" atau projek Genom manusia, dimana mereka mendata semua DNA di tubuh manusia dan membandingkannya dengan DNA mahluk hidup lainnya.Selain dengan pisang, manusia juga memiliki kesamaan DNA dengan lalat buah sekitar 60%,tikus yang sekitar 75%, dan sekitar 80% dengan sapi. [7]

Disisi lain, muncul sebuah paradigma baru yang dikenal paradigma berorientasi objek, paradigma ini merupakan sebuah terobosan dengan melibatkan contoh dan objek. Paradigma ini muncul sebagai akibat ditemukannya kekurangan dari paradigma terdahulunya yaitu paradigma terstrukur. Sehingga konsep objek memberikan kemudahan untuk penggunaan ulang kode yang telah dituliskan (*reuseable code*).

Bambang Hariyanto, Ir., M.T. dalam bukunya menyatakan bahwa Paradigma Berorientasi Objek adalah:

"*...cara pandang yang bukan sekedar algoritma yang diterapkan pada bahasa berorientasi objek. Cara pandang ini harus dibiasakan dari level mikro dan level tertinggi. Cara berpikir berorirentasi objek adalah segala sesuatu dipandang sebagai objek...*" [4]

Seperti yang diuraikan dari Bambang Hariyanto, Ir., M.T., mengenai paradigma berorientasi objek maka setiap mahluk hidup terutama manusia dapat dipandang sebagai komponen yang terdiri dari DNA, sehinga cara pandang ini dilihat sebagai sebuah objek. Seperti yang telah disampaikan diatas, bahwa setiap DNA yang dimiliki oleh tiap mahluk hidup baik manusia, hewan atupun tumbuhan akan memiliki pengaruh yang sangat signifikan untuk perkembangan di dalam tiap mahluk hidup baik secara sifat atupun bentuknya. Oleh karenanya struktur keluarga yang terdiri dari ayah dan ibu dapat ditelusuri untuk sifat yang dimiliki oleh anaknya.

Berdasarkan pemaparan diatas maka dapat dilakukan identifikasi permasalahan yaitu :

a. DNA yang dimiliki setiap mahluk hidup seperti manusia dan hewan memiliki pengaruh besar terhadap turunannya untuk melakukan pewarisan baik sifat ataupun bentuknya terhadap turunannya. Sehingga untuk bisa mengetahui sifat / bentuk yang dimiliki seorang anak dapat ditelusuri berdasarkan sifat / bentuk induknya dari kelas ayah dan ibunya, konsep inilah yang dikenal dengan *inheritance* atau pewarisan.
b. Seorang anak dapat dipandang sebagai objek karena memiliki ciri pembeda (*atributte*) dan memiliki kemampuan untuk melakukan sesuatu (*methode/operation*).
c. Untuk dapat menelusuri penurunan tiap mahluk, maka perlu paradigma berorientasi objek karena hal ini tidak dapat ditelusuri dengan paradigma terstruktur.
d. Untuk mengkaji dan menelurusi konsep *multy inheritance* maka perlu diimplementasikan berdasarkan teori dan metode yang ada.

Berdasarkan pemaparan diatas maka penulis akan membahas tentang "***Multy Inheritance* dalam Paradigma Berorientasi Objek dengan menggunakan metode Coad Yourdon".**

Batasan Masalah

Untuk memperjelas pembahasan yang tengah dikaji, maka penulis perjelas dengan ruang lingkup seperti :

1. *Multi inheritance* ini diterapkan pada kelas keluarga dengan struktur pewarisan berganda bertingkat yang terdiri dari kakek dari ayah, nenek dari ayah, kakek dari ibu, nenek dari ibu, ayah, ibu dan anak. Konsep pewarisan inilah akan ditelusuri penurunan sifat dari masing – masing objek yang menjadi pewaris untuk kelas dibawahnya.
2. Metode analisis dan desain menggunakan konsep berorientasi objek dari Coad Yourdon
3. Notasi pemodelan yang digunakan yaitu notasi UML (*unified modelling language).* Bahasa pemrograman yang digunakan untuk menerapkan konsep berorientasi objek ini adalah bahasa pemrograman C++.
4. Penentuan ciri / sifat (*attribute*) yang pasti dimiliki oleh kelas turunan berdasarkan hasil penurunan dari tiap kelas induknya saja, sehingga kelas yang memiliki atribut yang sama akan dikelompokkan atau disatukan ke dalam kelas yang sama.
5. Dua kelas yang memiliki penurunan dari satu kelas induk yang sama (pewarisan tunggal) tidak dapat diwariskan kembali ke dalam satu kelas anak (pewarisan ganda) dikarenakan kelas anak *ambigous* terhadap kelas induknya.

Tujuan Penelitian

Tujuan dari penelitian ini yaitu mengimplementasikan konsep *multi inheritance* dari kelas induk ke kelas turunannya sehingga dapat ditelusuri dan dibandingkan hasil pewarisan secara bertingkat dari tiap kelas induk ke kelas anak.

Manfaat Penelitian

Manfaat penelitian yang diharapkan dapat tercapai diantaranya adalah :

1. Berdasarkan struktur kelas, dapat dijadikan sebagai template untuk mengkaji penelurusan penurunan dari tiap kelas induk ke kelas turunannya.
2. Dapat dijadikan acuan untuk pengambilan keputusan untuk mengambil sifat / ciri yang pasti dmiliki dari kelas turunan sebuah kelas berdasarkan metode berorientasi objek.

Metodologi Penelitian

Metode penelitian yang digunakan mengikuti metode pengembangan sistem yaitu waterfall. Secara rinci metode / tahapan yang dilakukan adalah sebagai berikut :

1. Analisis (Analysis)
   Pada tahap ini penulis melakukan analisa sesuai metode yang diusulkan oleh Coad Yourdon yaitu identifikasi terhadap Kelas dan Objek *(Identyfing Class and Object)*, identifikasi Struktur *(Identyfing Structure)*, identifikasi Subjek *(Identyfing Subject)*, identifikasi Atribut *(Identyfing Attribute)* dan identifikasi Operasi *(Identyfing Operation)* yang kemudian diterapkan menjadi struktur diagram kelas lengkap dengan atribut, operasi dan strukturnya.

2. Perancangan (Desain)
   Pada tahap ini penulis melakukan perancangan sesuai tahapan yang diusulkan oleh Coad Yourdon, maka pada tahap ini penulis melakukan peracangan seperti membuat prototype domain masalah *(Problem Domain Component (PDC)),* rancangan interaksi manusia *(Human Interaction Component (HIC),* rancangan manajemen data *(Data Manegement Component (DMC),* rancangan *m*anajemen tugas *(Task Management Component (TMC)).*

3. Pembuatan kode program (Code)
   Pada tahap ini penulis menuliskan kode – kode dalam bahasa pemrograman bahasa C++, struktur penulisan kelas yang dibuat sesuai dengan tahap perancangan.

4. Tes hasil kode
   Pada tahap ini penulis melakukan pengujian terhadap kode – kode program yang dituliskan, metode yang digunakan yaitu menggunakan metode white box. Langkah yang dilakukan yaitu dengan cara menekan Control-F9, sehingga comiler dari bahasa pemrograman C++ melakukan pengetesan tiap kode program yang dituliskan. Penulis melakukan penegcekan terhadap logika dan sintak dari bahasa pemrograman C++ tersebut.

PEMBAHASAN

Untuk mengkaji permasalahan diatas maka perlu beberapa dukungan teori seperti tentang metode berorientasi objek dari Coad Youdon. Metode berorientasi objek akan selalu terkait dengan istilah objek, dimana segala permasalahan didunia nyata terdiri dari berbagai objek, yang nantinya akan direpresentasikan dalam sebuah kelas. Sehingga untuk mengkaji metode ini yang harus dilakukan pertama kali adalah mengidentifikasi kelas dan objek.

Menurut Sariadin, dalam pemrograman berorientasi objek setiap objek akan memiliki data (vaiable / konstanta) adan method (perilaku/kemampuan melakukan suatu fungsi). Jadi objek dapat didefinisikan sebagai entitas yang memiliki data dan method [2]. Sementara class adalah fasilitas untuk membuat objek tersebut.

Untuk lebih jelasnya, apa itu kelas maka sebenarnya class bukanlah objek, class adalah blue print dari objek sehingga class merupakan fasilitas untuk membuat objek yang didalamnya mengandung informasi tentang data (atribut) dan perilaku (prosedur) dari sebuah objek. Lalu bagaimana dengan objek sendiri? Menurut *webster dictionary* objek dapat didefinisiskan sebagai : "*a person or thing to wich action*" dan objek pun diartikan oleh Pooley R bahwa : "*A thing yo can interact with*". [3], yang memiliki ciri seperti : *Concrete, Roles, Relational, Event dan displayable*.

Metode berorientasi objek pun memiliki kelebihan dibandingkan metode terstruktur, hal ini seperti yang diungkapkan oleh Adi N, ST., MMSI. [1] yaitu :

1. Pendekatan lebih pad data bukan pada prosedur atau fungsi
2. Program besar dibagi pada apa yang dinamakan objek-objek
3. Struktur data dirancang dan menjadi karakteristik dari obje-objek
4. Fungsi – fungsi yang mengoperasikan data tergabung dalam suatu objek yang sama
5. Data tersembunyi dan terlindung dari fungsi / prosedur yang ada diluar
6. Objek – objek dapat saling berkomunikasi dengan saling mengirim pesan satu sama lain
7. Pendekatan adalah dari bawah ke atas (*bottom up approach*)

Metode Coad Yourdon diambil dari dua nama pengusulnya yaitu Coad Untuk mengkaji metode berorinetasi objek dari Coad Yourdon maka Coad dan Yourdon mengusulkan 5 langkah pada tahap analisis seperti [5] :

1. Identifikasi Objek dan Kelas.
   Identifikasi Objek dan Kelas objek objek yaitu mengidentifikasi apa saja yang termasuk kedalam objek dan kelas yang berkaitan. Sehingga dalam sebuah kelas menyimpan objek-objek yang berkaitan.

2. Identifikasi Struktur
   Identifikasi Struktur yaitu membuat pola keterkaitan antar tiap kelas dan objek yang telah diidentifikasi sebelumnya. Struktur terbagi menjadi 2 macam yaitu :

   a. *Gen-Spec* merupakan bagian keterhubungan antar kelas yang memiliki relasi berupa '*is-a'*
   b. *Whole-Part* mempunyai relasi berupa '*has-a*'. contohnya : sebuah **mobil** mempunyai **roda**, **mesin** , **chasis** …dsb.

3. Identifikasi Subjek
   Subjek adalah sebuah mekanisme yang menunjukkan sintak atau semantic suatu domain masalah yang komplek. Masing-masing kelas dan objek dalam model objek menunjukkan ke subjeknya.

4. Identifikasi Atribut
   Atribut merupakan item data yang terdapat pada setiap objek atau kelas. Setiap objek mempunyai range untuk menyatakan batasan objek lainnya.

5. Identifikasi Operasi
   Sebuah operasi menyatakan tingkah laku atau kebiasaan yang dilakukan oleh objek atau kelas yang dapat dijadikan panduan untuk mengukur kemampuan proses dari suatu objek atau kelas.

   Lalu mengusulkan 4 langkah pada tahap desain yaitu :

1. Identifikasi domain masalah (*Problem domain components* ((PDC)
   Tahap ini menjelaskan inti permasalahan yang sedang diamati. Hasil dari tahapan analisis yang dilakukan dapat secara langsung digunakan ataupun kalau akhirnya ditemukan sesuatu yang baru dapat ditambahkan pada langkah ini.

2. Identifikasi manusia komputer (*Human Interactions Components* ((HIC)
   Menjelaskan rancangan interaksi antara aplikasi yang akan dibuat dengan user sebagai pengguna. Rancangan ini memfokuskan pada menu / modul dan perancangan antar muka berupa *form / report.*

3. Identifikasi alur kerja (*Task management Components* ((TMC)
   Tahap ini menjelaskan rancangan tugas (prosedur kerja) dari masing-masing komponen yang terkait dalam masalah yang sedang diamati.

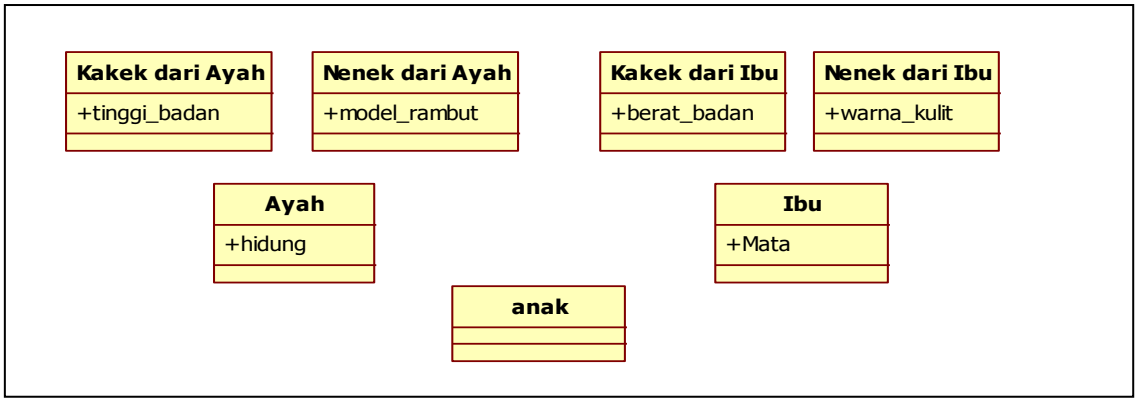
4. Identifikasi data (*Data Management Components* ((DMC)
   Tahap ini melakukan perancangan data. Data yang dimaksud berupa tabel, file, record serta spesifkasi penjelasan antar data tersebut. Data ini akan diperoleh dari atribut yang sudah brhasil diidentifikasi dari tahap analisis sebelumnya.

Tahapan analisis

Seperti yang telah diperjelas diatas, maka pada tahap ini akan diidentifikasi objek dan kelas yang menjadi domain permasalahan dari struktur kelas keluarga yang terdiri dari kakek dari ayah, nenek dari ayah, kakek dari ibu,nenek dari ibu, ayah, ibu dan anak. Tiap tahapan ini mengikuti model dari Coad Youdon yang mengidentifikasi 5 langkah seperti berikut ini :

1. Identifikasi kelas dan objek

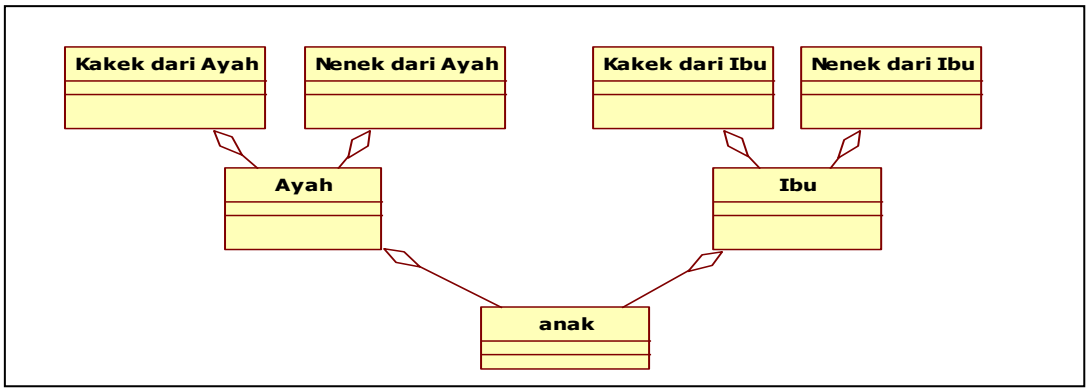
Identifikasi objek dan kelas menjadi satu tahapan, dikarenakan kelas dan objek sangat berkaitan, jika objek sudah ditemukan maka kelas pun dapat diidentifikasi. diidentifikasi direpresentasikan dalam kelas keluarga seperti gambar 1 berikut ini :



Gambar 1. Identifikasi kelas dan Objek

2. Identifikasi struktur

Dengan melakukan identifikasi struktur dari kelas keluarga maka akan menampilkan pola yang menghubungkan kelas satu dengan kelas lainnya, terutama kelas induk dengan kelas anaknya. Struktur ini memiliki pola *whole part* atau bagian dari maksudnya adalah bahwa dalam sebuah keluarga terdiri dari kelas-kelas seperti : kakek dari ayah, nenek dari ayah yang diwariskan ke kelas ayah, lalu terdiri dari kakek dari ibu, nenek dari ibu yang diwariskan ke kelas ibu. Lalu kelas ayah dan ibu diwariskan ke kelas anak. Gambar berikut berhasil mengidentifikasi struktur dari kelas keluarga seperti berikut ini :



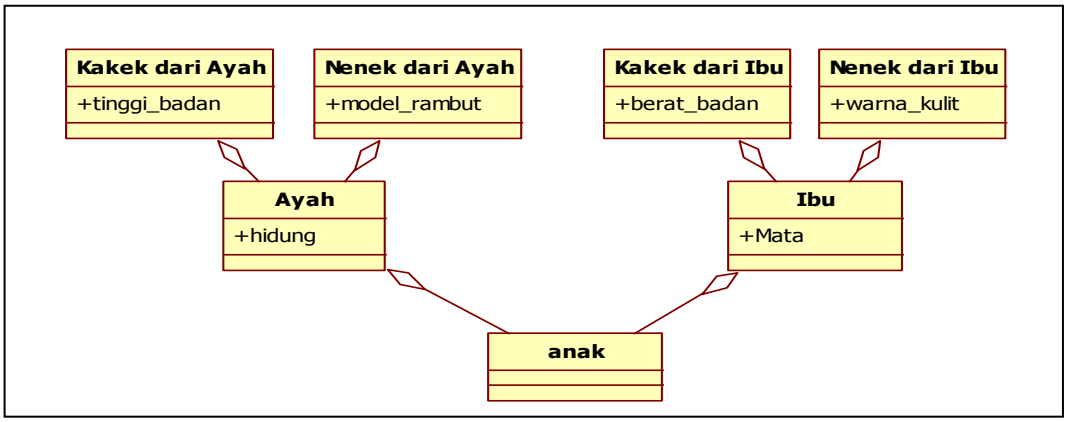
Gambar 2. Identifikasi Struktur

3. Identifikasi subjek

Identifikasi subjek dilakukan dengan asumsi kajian yang dibahas pada penelitian memiliki lingkup yang luas dengan domain permasalahan yang menampilkan banyak objek. Dikarenakan struktur dari kelas yang dibahas, memiliki lingkup yang kecil, sehingga pada tahap ini tidak dilakukan.

4. Identifikasi atribut

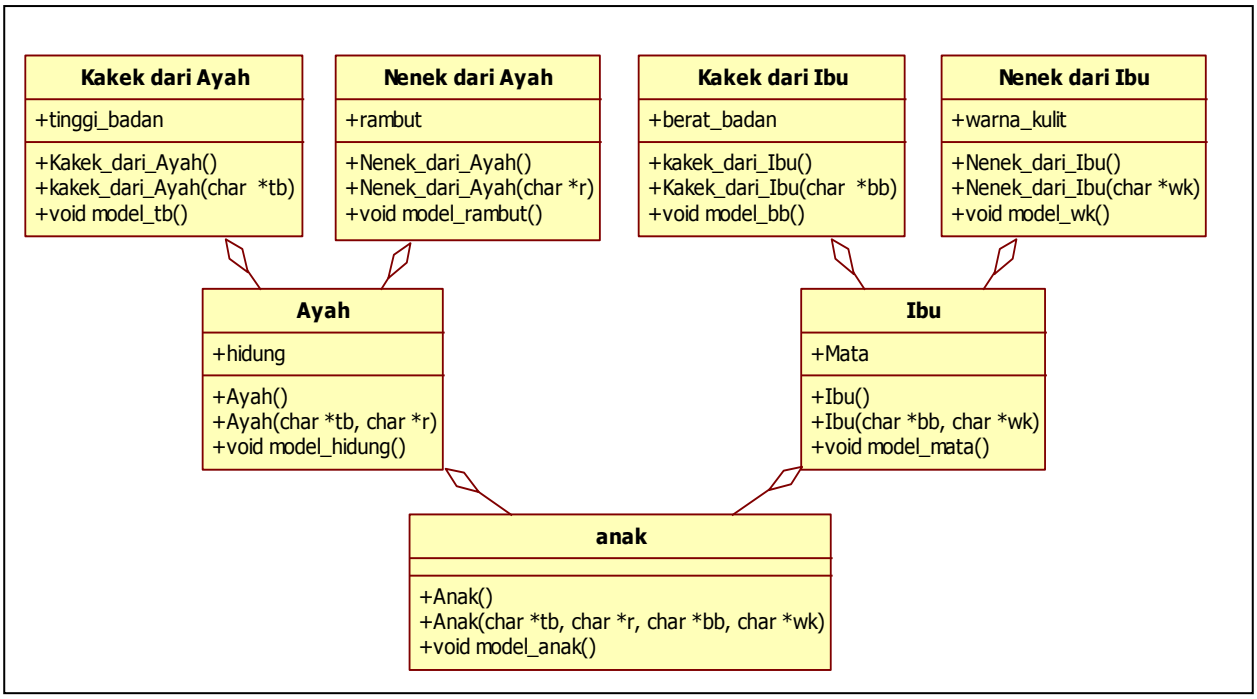
Setiap kelas yang telah teridentifikasi maka akan memiliki atribut pembeda dengan kelas lainnya, sehingga jika sebuah kelas memiliki atribut yang sama tidak akan ditulis lagi dikelas yang selevel (sub kelas) namun akan dituliskan dikelas diatasnya (induk kelas). Identifikasi atribut pada kelas keluarga ditampilkan seperti gambar berikut ini :



Gambar 3. Identifikasi Atribut

5. Identifikasi layanan (*method*)

Identifikasi layanan dilakukan untuk mengetahui kemampuan dari tiap kelas yang sudah berhasil diidentifikasi. Hasil gambar berikut ini menampilkan layanan dari kelas keluarga seperti berikut ini :

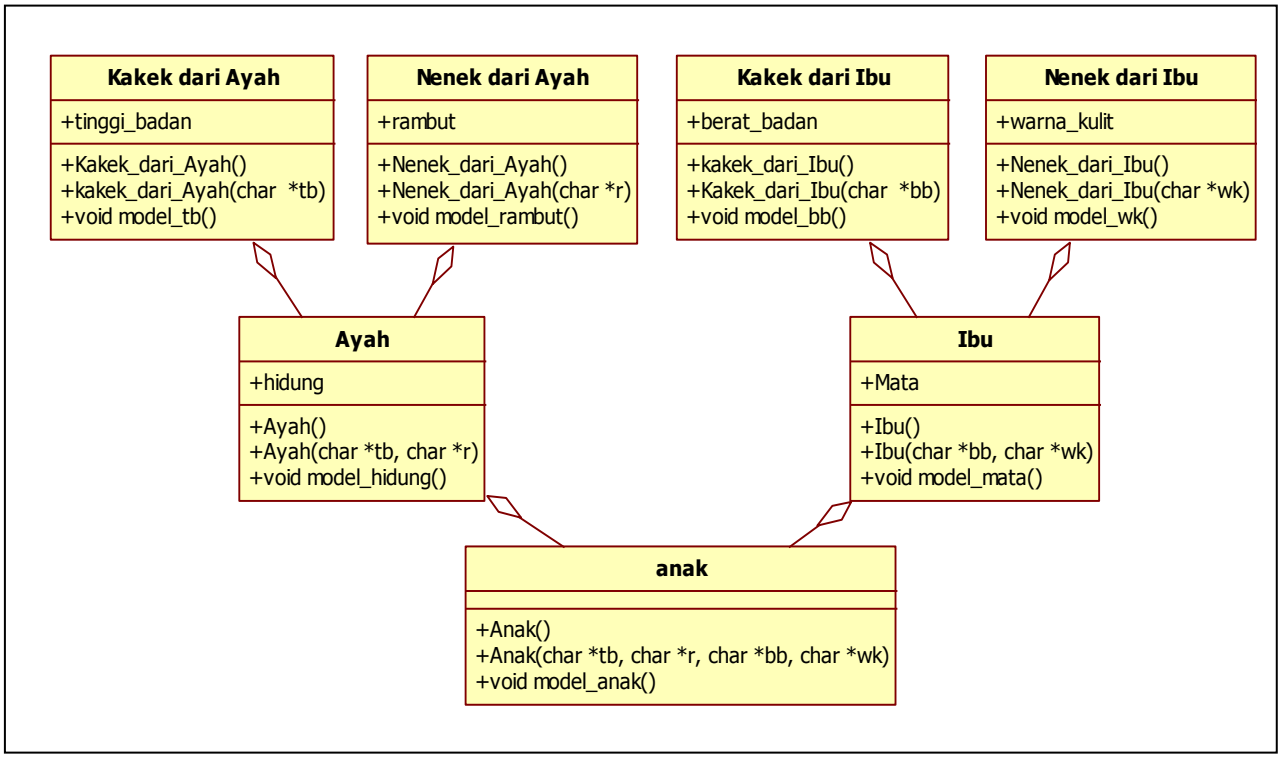


Gambar 4. Identifikasi Layanan

Tahapan Perancangan

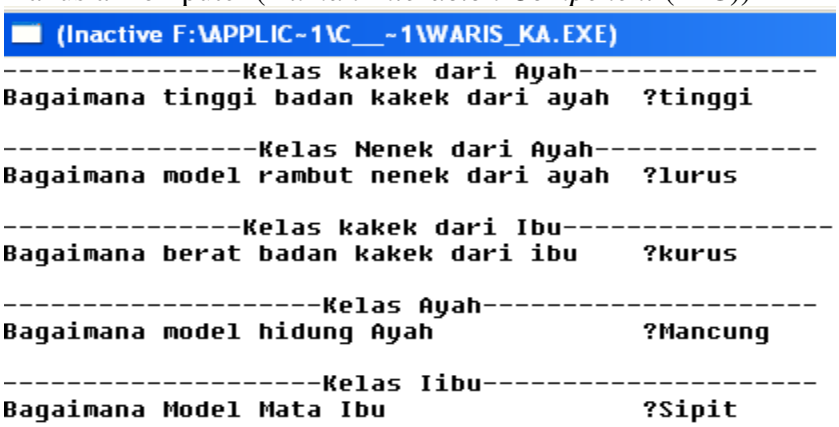
1. Identifikasi domain masalah (*Problem Domain Component* (PDC))

Berdasarkan hasil akhir dari tahapan analisis yang dilakukan maka diperoleh bahwa hasil identifikasi kelas dapat digambarkan seperti berikut ini :

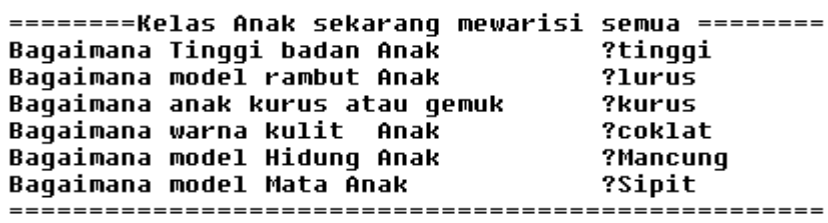


Gambar 5. Identifikasi Domain Masalah

2. Identifikasi Manusia Komputer (*Human Interacion Compenent* (HIC))



Gambar 6. Form Masukan Kelas Keluarga



Gambar 7. Form Keluaran Kelas Anak

3. Identifikasi Alur Kerja (*Task Management Component* (TMC))
   Identifikasi alur kerja meliputi prosedur atau fungsi yang telah berhasil diidentifikasi pada tahap analisis yaitu tahap identifikasi layanan (Method). Tahap ini lebih dikenal dengan algoritma. Hasil identifikasi seperti berikut ini :
   1. Class Kakek_dari_Ayah(), Class ini memiliki 3 prosedur seperti berikut ini :

{*prosedur berupa konstruktor yang melakukan inisialisasi tinggi badan*}

Algoritma Kakek_dari_Ayah() //

Deklarasi tinggibadan : char[20] //*tb adalah tinggi badan bertipe char space 20*

Deskripsi

tb =”tinggi badan kakek dari ayah…..” write(tb)

{*prosedur dengan parameter yang mengisi tb dengan keterangan tinggi badan*}

Algoritma Kakek_dari_Ayah(char * tb)

Deklarasi tinggibadan : char[20] //*tb adalah tinggi badan bertipe char space 20*

Deskripsi

read (tinggibadan, tb)

Algoritma model_tb() // *prosedur menampilkan keterangan tinggi badan*

Deklarasi tinggibadan : char[20] //*tb adalah tinggi badan bertipe char space 20*

Deskripsi

write(“Tinggi badan kakek adalah :”,tb)

2. Class Nenek_dari_Ayah(), Class ini memiliki 3 prosedur seperti berikut ini :

```
{prosedur berupa konstruktor yang melakukan inisialisasi model rambut}
Algoritma  Nenek_dari_Ayah()
    Deklarasi    rambut : char[20]  //rambut bertipe char space 20
    Deskripsi
        rambut =”rambut nenek dari ayah…..”  write(rambut)
{prosedur dengan parameter yang mengisi tipe data rambut dengan r }
Algoritma Nenek_dari_Ayah(char * r)
    Deklarasi  rambut : char[20]  //rambut adalah r bertipe char space 20
    Deskripsi
        read (rambut,r)
Algoritma model_rambut() // prosedur menampilkan keterangan rambut
    Deklarasi   rambut : char[20]  //tb adalah rambut bertipe char space 20
    Deskripsi
          write(“Model rambut adalah     :”,r)
```

3. Class Kakek_dari_Ibu(),Class ini memiliki 3 prosedur seperti berikut ini :

```
{prosedur berupa konstruktor yang melakukan inisialisasi berat  badan}
Algoritma  Kakek_dari_Ibu() //
    Deklarasi   beratbadan : char[20]  //tb adalah berat badan bertipe char space 20
    Deskripsi
          bb =”berat badan kakek dari ibu…..”        write(bb)
{prosedur dengan parameter yang mengisi bb dengan keterangan berat  badan}
Algoritma Kakek_dari_Ibu(char * bb)
    Deklarasi    beratbadan :char[20]  //bb adalah berat  badan bertipe char space 20
    Deskripsi
          read (beratbadan,bb)
Algoritma model_bb() // prosedur menampilkan keterangan berat badan
    Deklarasi    beratbadan : char[20]  //bb adalah berat badan bertipe char space 20
    Deskripsi
          write(“Berat badan kakek adalah     :”,bb)
```

4. Class Nenek_dari_Ibu(), Class ini memiliki 3 prosedur seperti berikut ini :

```
{prosedur berupa konstruktor yang melakukan inisialisasi berat  badan}
Algoritma  Nenek_dari_Ibu() //
    Deklarasi   warnakulit : char[20]  // warna kulit bertipe char space 20
    Deskripsi
          wk =”warna kulit nenek dari ibu…..”  write(wk)
{prosedur dengan parameter yang mengisi wk dengan keterangan warna kulit}
Algoritma Nenek_dari_Ibu(char * wk)
    Deklarasi  warnakulit : char[20]  //wk adalah warna kulit bertipe char space 20
    Deskripsi
          read (warnakulit,wk)
Algoritma model_wk() // prosedur menampilkan keterangan warna kulit
    Deklarasi warnakulit : char[20]  //wk adalah warna kulit bertipe char space 20
    Deskripsi
          write(“Warna kulit nenek  adalah     :”,wk)
```

5. Class Ayah(), Class ini memiliki 3 prosedur seperti berikut ini :

```
{prosedur konstruktor yang melakukan inisialisasi ket tinggi badan & hidung}
Algoritma  Ayah() //
    Deklarasi   hidung : char[20]  // hidung bertipe char space 20
    Deskripsi
          h =”hidung Ayah…..”  write(h)
{prosedur dengan parameter yang mengisi h dengan keterangan hidung}
Algoritma Ayah(char * h)
    Deklarasi    hidung  : char[20]  //h adalah hidung bertipe char space 20
    Deskripsi
          read (hidung,h)
Algoritma model_hidung() // prosedur menampilkan keterangan hidung
    Deklarasi  hidung : char[20]  //h adalah hidung bertipe char space 20
    Deskripsi
                write(“Model hidung ayah  adalah     :”,h)
```

6. Class Ibu(), Class ini memiliki 3 prosedur seperti berikut ini :

```
{prosedur konstruktor yang melakukan inisialisasi keterangan mata}
Algoritma  Ibu() //
    Deklarasi   mata : char[20]  // mata bertipe char space 20
    Deskripsi
            m =”Mata Ibu…..”  write(m)
{prosedur dengan parameter yang mengisi m dengan keterangan mata}
Algoritma Ayah(char * m)
    Deklarasi  mata  : char[20]  //h adalah mata bertipe char space 20
    Deskripsi
            read (mata,m)
Algoritma model_mata() // prosedur menampilkan keterangan mata
    Deklarasi   mata : char[20]  //h adalah mata bertipe char space 20
    Deskripsi
             write(“Model mata nenek  adalah     :”,m)
```

7. Class Anak(), Class ini memiliki 3 prosedur seperti berikut ini :

```
Algoritma  Anak() // {prosedur konstruktor yang inisialisasi ket pd atribut}
Deskripsi
        tb =”Model Tinggi badan Anak ..……” r   =”Model Rambut Anak ..……”
        bb =”Model Berat badan Anak ..……” wk=”Model Warna kulit Anak ..……”
        h  =”Model Hidung Anak ..……”         m =”Model Mata Anak ..……”
        write(m) write (tb)  write (r) write (bb)  write (wk)  write (h)  write (m)
{prosedur dengan parameter yang mengisi variable dengan keterangan atribut}
Algoritma Anak(char *tb, char *r, char *bb, char wk, char *h, char *m)
Deskripsi
        read (tinggibadan,tb) read (rambut,r) read (beratbadan,bb)
        read (warnakulit,wk)read (hidung,h)read (mata,m)
```

Algoritma model_Anak() // *prosedur menampilkan keterangan mata*

Deskripsi

write(“Model tinggi badan anak adalah :”,tb)

write(“Model rambut badan anak adalah :”,r)

write(“Model berat badan anak adalah :”,bb)

write(“Model warna kulit anak adalah :”,wk)

write(“Model hidung anak adalah :”,h)

write(“Model mata anak adalah :”,m)

4. Identifikasi Data (*Data management Component* (DMC))

Identifikasi data telah diperoleh dari tahap analisis pada identifikasi atribut. Selengkapnya dapat dilihat pada tabel beriktu ini :

Tabel 1. Identifikasi Data

No	Kelas	Atribut	Tipe	Space	Keterangan
1	KakekdariAyah	Tinggibadan	Char	20	
2.	NenekdariAyah	Rambut	Char	20	
3.	KakekdariIbu	Beratbadan	Char	20	
4.	NenekdariIbu	Warnakulit	Char	20	
5.	Ayah	Hidung	Char	20	
6.	Ibu	Mata	Char	20	
7.	Anak	Tinggibadan, Rambut, Beratbadan, Warnakulit, Hidung, Mata			Mewarisi atribut yang dimiliki dari kelas sebelumnya

Pembuatan Kode Program

```
// nama program : kelas Multy Inheritance
#include<iostream.h>
#include<string.h>
class kakek_dari_ayah{
protected : char tinggibadan[20];  //atribut
public : kakek_dari_ayah()  //method
            { strcpy(tinggibadan,"Tinggi badan kakek ayah biasa aja"); }

         kakek_dari_ayah(char *tb)
```

```
        { strcpy(tinggibadan,tb); }
      void modeltinggibadan()
        {cout<<"Bagaimana tinggi badan kakek dari ayah?"<<tinggibadan<<"\n"; }};



class nenek_dari_ayah {

protected : char rambut[20]; //atribut

public     : nenek_dari_ayah()    //method

        { strcpy(rambut,"Rambut nenek dari ayah yaaa... begitu"); }
      nenek_dari_ayah(char *r)
        { strcpy(rambut,r); }
      void modelrambut()
        {cout<<"Bagaimana model rambut nenek dari ayah ?"<<rambut<<"\n"; }};

class kakek_dari_ibu {

protected : char beratbadan[20]; //atribut

public     :  kakek_dari_ibu()//method

        { strcpy(beratbadan,"Berat badan kakek ibu biasa aja");}
      kakek_dari_ibu(char *bb)
        { strcpy(beratbadan,bb); }
      void modelberatbadan()
        { cout<<"Bagaimana berat badan kakek dari ibu  ?"<<beratbadan<<"\n"; } };

class nenek_dari_ibu {

protected : char warnakulit[20];  //atribut

public    : nenek_dari_ibu()  //method

        { strcpy(warnakulit,"warna kulit nenek dari ibu .....hihi..");}
      nenek_dari_ibu(char *bb)
        { strcpy(warnakulit,bb); }
      void modelwarnakulit()
        { cout<<"Bagaimana earna kulit nenek dari ibu ?"<<warnakulit<<"\n"; }};

class ayah {

protected : char hidung[20]; //atribut

public     : ayah()  //method

        { strcpy(hidung,"Model Hidung ayah biasa aja"); }
      ayah(char *H)
        { strcpy(hidung,H); }
      void ModelHidung()
```

```
                {cout<<"Bagaimana model hidung Ayah           ?"<<hidung<<"\n";  };
class ibu {

protected : char mata[20];  //atribut

public      : ibu() //method

                { strcpy(mata,"Model mata ibu biasa aja");}
              ibu(char *M)
                { strcpy(mata,M); }
              void ModelMata()
                { cout<<"Bagaimana Model Mata Ibu              ?"<<mata<<"\n"; } };

class anak : public kakek_dari_ayah,public nenek_dari_ayah, public kakek_dari_ibu,public
nenek_dari_ibu, public ayah, public ibu {

public      : anak()

                { strcpy(tinggibadan,"tinggi badan anak");
                  strcpy(rambut,"rambut anak");
                  strcpy(beratbadan,"berat badan anak");
                  strcpy(warnakulit,"warna kulit anak");
                  strcpy(hidung,"hidung anak");
                  strcpy(mata,"mata anak"); }
             anak(char *tb,char *r, char *bb,char * wk,char *H,char *M)
                { strcpy(tinggibadan,tb);
                  strcpy(rambut,r);
                  strcpy(beratbadan,bb);
                  strcpy(warnakulit,wk);
                  strcpy(hidung,H);
                  strcpy(mata,M); }
             void ModelHidungMataAnak()
                  { cout<<"Bagaimana Tinggi badan Anak         ?"<<tinggibadan<<"\n";
                    cout<<"Bagaimana model rambut Anak         ?"<<rambut<<"\n";
                    cout<<"Bagaimana anak kurus atau gemuk     ?"<<beratbadan<<"\n";
                    cout<<"Bagaimana warna kulit  Anak         ?"<<warnakulit<<"\n";
                    cout<<"Bagaimana model Hidung Anak         ?"<<hidung<<"\n";
                    cout<<"Bagaimana model Mata Anak           ?"<<mata<<"\n"; } };
      main() {
        cout<<"----------Kelas kakek dari Ayah----------"<<"\n";
        kakek_dari_ayah A("tinggi");
        A.modeltinggibadan(); cout<<"\n";
        cout<<"----------Kelas Nenek dari Ayah----------"<<"\n";
        nenek_dari_ayah B("lurus");
        B.modelrambut(); cout<<"\n";
        cout<<"----------Kelas kakek dari Ibu----------"<<"\n";
        kakek_dari_ibu C("kurus");
        C.modelberatbadan(); cout<<"\n";
        cout<<"----------Kelas Ayah----------"<<"\n";
        ayah D("Mancung");
        D.ModelHidung(); cout<<"\n";
        cout<<"----------Kelas Iibu----------"<<"\n";
```

```
ibu E("Sipit");
E.ModelMata(); cout<<"\n"; cout<<"\n"; cout<<"\n";
cout<<"========Kelas Anak sekarang mewarisi semua ========"<<"\n";
anak F("tinggi","lurus", "kurus","coklat","Mancung", "Sipit");
F.ModelHidungMataAnak();
cout<<"================================================="<<"\n";
return 0; }
```

KESIMPULAN DAN SARAN

Kesimpulan

1. Hasil penurunan pewarisan ini memiliki struktur bertingkat dengan hasil bahwa kelas anak memiliki semua atribut / ciri yang dimiliki oleh kelas diatasnya.
2. Atribut yang dimiliki oleh kelas diatas, pasti dimiliki oleh kelas dibawahnya, namun penurunan tersebut tidak dapat diturunkan dari kelas yang sama dengan atribut yang berbeda karena akan menyebabkan ambigu pada kelas anak, sehingga penurunan sifat dari konsep DNA tidak bisa diterapkan pada konsep berorientasi objek.

Saran

Untuk pengembangan hasil metode berorientasi objek dengan lebih optimal maka perlu pengembangan keilmuan yang dilakukan terus agar dapat menyelesaikan permasalahan tersebut.

DAFTAR PUSTAKA

[1] *Adi Nurgtoho, ST., MMSI, Pemrograman Berorientasi Objek, Penerbit Informatika Bandung 2004*

[2] Sariadin Siallagan, Pemrograman Java, Penerbit Andi Yogyakarta 2009

[3] Stevens, P., with Pooley R, "*Using UML Software Engineering with Objects and Components*" (updates edition), Addision Wisley, Harlow., 2000

[4] Bambang Hariyanto, Ir., M.T., Rekayasa Perangkat Lunak dengan metode Berorientasi Objek, Penerbit Informatika, Bandung, 2004.

[5] Object Oriented Analysis (OOA) dan Object Oriented Design(OOD) Peter Coad dan Edward Yourdan, 1990

[6] http://www.academia.edu/6355553/PERBEDAAN_SEL_PROKARIOTIK_DAN_EUKARIOTIK diakses terakhir Februari 2014

[7] http://www.kaskus.co.id/thread/5336801e17cb17e5188b47ba/fakta-unik-dna-manusia Akses terakhir Februari 2014