DAFTAR ISI

[Halaman ini sengaja dibiarkan kosong]

KATA PENGANTAR

Praktikum Pemrograman Berorientasi Objek dilaksanakan bersamaan dengan mata kuliah Pemrograman Berorientasi Objek di semester dua yang membahas tentang Object Oriented Programming. Praktikum ini tidak memiliki beban sks tersendiri untuk aktivitas di laboratorium, melainkan menjadi 1 paket dengan mata kuliahnya. Adapun kegiatan di laboratorium jumlahnya setara dengan 2 jam tatap muka.

Di dalam setiap modul diberikan ringkasan dasar teori dilengkapi dengan analisa dan contoh implementasi program yang diharapkan bisa membantu mahasiswa untuk memahami konsep serta memudahkan pengerjaan tugas praktikum. Di akhir bagian tiap modul diberikan juga beberapa contoh soal praktikum untuk dijadikan acuan bagi asisten di laboratorium.

Akhir kata, modul ini masih perlu untuk terus diperbaiki. Penulis sangat berterima kasih atas masukan dan saran yang diberikan.

Surabaya, Februari 2016

Penulis

[Halaman ini sengaja dibiarkan kosong]

TUJUAN INSTRUKSIONAL KHUSUS:

Mahasiswa mampu menganalisis studi kasus yang diberikan dan kemudian mampu membuat solusi pemrograman dengan menggunakan konsep object-oriented.

KOMPOSISI PENILAIAN:

Pretest/Tes Awal : 20 poin
Kedisiplinan : 20 poin
Laporan Resmi : 20 poin
Praktek : 40 poin

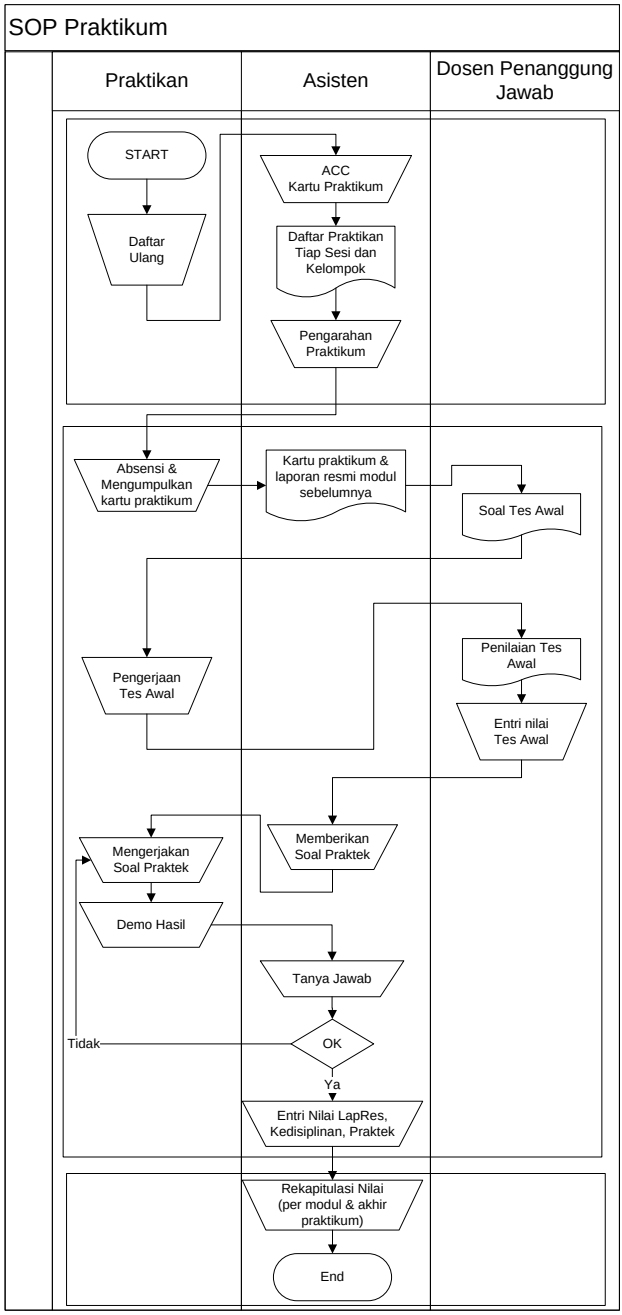
ALOKASI WAKTU PRAKTIKUM:

Total waktu: 120 menit

- 10 menit : persiapan
- 15 menit : tes awal
- 10 menit : penjelasan modul
- 60 menit : praktek
- 25 menit : presentasi dan tanya jawab hasil praktikum oleh asisten

[Halaman ini sengaja dibiarkan kosong]

PROSEDUR PELAKSANAAN :



KETERANGAN :

1. Praktikan yang datang **TERLAMBAT** (sebelum praktek dimulai) diperbolehkan mengikuti praktikum tetapi **kehilangan POIN Tes awal dan maksimum mendapatkan 10 POIN Kedisiplinan (nilai maksimum yang bisa didapat adalah 70)**. Asisten praktikum akan memberi keterangan pada kartu biru praktikan bahwa untuk modul tersebut adalah "**TERLAMBAT**".
2. Jika mahasiswa berhalangan hadir, maka **WAJIB** mengurus **PRAKTIKUM SUSULAN** ke Dosen Penanggung Jawab. Ijin **praktikum susulan** akan diberikan **hanya jika** praktikan menyerahkan ***surat keterangan*** yang jelas kepada Dosen Penanggung Jawab. Surat keterangan dapat berupa surat dokter atau surat keterangan yang ditandatangani orangtua/wali. **Pemberian ijin praktikum susulan maksimal hanya 2 (dua) modul** untuk setiap praktikum dalam 1 semester**.** Toleransi permintaan praktikum susulan adalah ***selambat-lambatnya 3 hari*** setelah **jadwal modul yang seharusnya** atau selambat-lambatnya **3 hari sebelum modul berikutnya**. Dan praktikan tersebut **TIDAK** mendapatkan **POIN Pretest/Tes Awal dan 5 POIN Kedisiplinan (nilai maksimum yang bisa didapat adalah 65)**. Dosen penanggung jawab praktikum akan memberi keterangan pada kartu biru praktikan bahwa untuk modul tersebut adalah "**SUSULAN**".
3. Dosen Penanggung Jawab atau asisten yang mewakili akan mengadakan *pretest* (tes awal) bagi praktikan pada jadwal sesi praktikum. Tes ini bersifat individu. Setelah itu dosen akan memasukkan nilai *pretest* ke **POIN Pretest/Tes Awal**.
4. Praktikan yang sudah menyelesaikan tugas praktikumnya lebih awal dari waktu yang disediakan bisa mengajukan permintaan demonstrasi hasil praktikum ke asisten dan setelah itu boleh meninggalkan tempat. Tetapi jika waktu yang disedikan telah habis maka demo hasil praktikum dilaksanakan saat itu juga tanpa ada perpanjangan waktu. Kemudian asisten akan memasukkan nilai untuk **POIN Hasil Praktikum**.
5. Praktikan **HARUS** mengumpulkan ***laporan resmi*** yang berisi kegiatan yang telah dilakukan di laboratorium yaitu soal praktikum beserta analisis masalah, hasil desain, hasil

demonstrasi, dan kesimpulan ke asisten ***selambat-lambatnya 2 (dua) hari*** sebelum praktikum modul berikutnya. Asisten akan memberikan penilaian untuk **POIN <u>Laporan Resmi</u>**. Praktikan yang sudah mengumpulkan ***laporan resmi*** modul sebelumnya ***ber-HAK*** untuk mengikuti praktikum di laboratorium sesuai jadwal yang telah ditetapkan. Praktikan yang tidak mengumpulkan laporan resmi tidak ***ber-HAK*** untuk mengikuti praktikum modul berikutnya.

[Halaman ini sengaja dibiarkan kosong]

STANDARDISASI RANGE NILAI

1. TES AWAL
   a. 0 – 5: KURANG
      Jawaban praktikan sama sekali tidak berhubungan dengan materi yang dibahas. Praktikan TIDAK BERHAK mengikuti praktikum khusus untuk modul yang sedang diujikan.
   b. 6 – 15 : SEDANG
      Jawaban praktikan sesuai dengan materi yang dibahas tetapi belum/ kurang lengkap.
   c. 16 – 20 : BAIK
      Jawaban praktikan cukup dan atau sudah sesuai dengan materi yang dibahas.

2. KEDISIPLINAN
   a. 0 – 10 : KURANG. Beberapa contoh:
      - Hadir terlambat tanpa alasan yang tepat atau berhalangan hadir.
      - Bermain (game) atau menjalankan aplikasi yang tidak berhubungan dengan praktikum selama di laboratorium.
      - Merubah setting komputer yang dipakai tanpa ijin administrator.
      - Mencontek pekerjaan kelompok lain.
      - Membuang sampah di dalam laboratorium.
   b. 11 – 15 : SEDANG. Beberapa contoh:
      - Ramai/ gaduh di dalam laboratorium.
      - Mengubah/ memindah posisi tempat duduk.
      - Berkomunikasi dengan kelompok lain.
   c. 16 – 20 : BAIK
      Menaati semua peraturan dan tata tertib praktikum selama di dalam laboratorium, serta **datang tepat waktu**.

3. HASIL PRAKTIKUM
   a. 0 – 17 : KURANG. Beberapa contoh:
      - Tugas praktikum tidak/belum selesai dikerjakan.
      - Praktikan tidak bisa menjawab pertanyaan asisten atau tidak bisa menjelaskan kesalahan hasil praktikumnya dengan benar.
   b. 18 – 31 : SEDANG. Beberapa contoh:

- Tugas praktikum sudah dikerjakan, tetapi hasilnya belum sesuai dengan yang dimaksud oleh soal.
- Praktikan bisa menjawab pertanyaan asisten atau mampu menjelaskan kesalahan hasil praktikumnya kepada Asisten.

c. 32 – 40 : BAIK. Dengan syarat
- Tugas praktikum dikerjakan sesuai dengan permintaan soal.
- Praktikan mampu menjawab pertanyaan Asisten dengan benar.

4. LAPORAN RESMI

a. 0 – 9 : KURANG. Beberapa contoh:
- Sama dengan laporan resmi praktikan yang lain baik sebagian bab maupun isi keseluruhannya.
- Terlambat mengumpulkan dari jadwal yang sudah ditentukan (satu pekan setelah praktikum).

b. 10 – 15 : SEDANG
- Ada format yang belum lengkap. Misal: Kurang Bab Pembahasan, kurang Hasil Analisis, Kesimpulan, dsb.
- Isi Laporan Resmi masih belum sesuai dengan tugas yang dikerjakan.
- Penjelasan korelasi antar bab belum baik.

c. 16 – 20 : BAIK
. Format Laporan Resmi lengkap.
a. Isinya sesuai dengan tugas yang dikerjakan.
b. Korelasi antar bab dijelaskan dengan baik.

FORMAT LAPORAN RESMI

- **setiap kelompok mengumpulkan 1 laporan resmi untuk setiap modul**
- Kertas HVS A4 dicetak dengan tinta hitam dan ***double-sided*** **(bolak-balik)**
- Margin atas dan luar 3 cm, sedangkan margin bawah dan dalam 2 cm.
- Font Arial ukuran 12 dengan spasi Single. Ukuran 14 untuk Header pokok bahasan.
- Staples 3 di sisi kiri tanpa dijilid.
- Isi Laporan Resmi :
    - Cover (lihat gambar di bawah)
    - Soal Tugas Praktikum
    - Analisis masalah
    - Hasil Praktikum (gambar/program dan penjelasannya)
    - Pembahasan (penjelasan kekurangan hasil praktikum ketika diuji oleh asisten dan bagaimana solusinya)
    - Kesimpulan

PRAKTIKUM STRUKTUR DATA

LAPORAN RESMI
MODUL 1 – KONSEP OOP DI RUBY

[LOGO UPN]

SESI/KELOMPOK : ……/ ……...
[NPM dan NAMA PRAKTIKAN 1]
[NPM dan NAMA PRAKTIKAN 2]

ASISTEN : .....
PARAF : …….
LABORATORIUM PEMROGRAMAN
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS PEMBANGUNAN NASIONAL ”VETERAN” JAWA TIMUR
2012

[Halaman ini sengaja dibiarkan kosong]

MODUL 1
CLASS DAN OBJECT, SERTA PERANCANGAN OBJECT ORIENTED

Class dan object adalah dasar dari paradigma pemrograman berorientasi obyek. Class adalah representasi sebuah tipe data abstrak (Abstract Data Type) yang di dalamnya terdapat struktur data dan operasi (fungsi) yang berkaitan dengan struktur data tersebut. Struktur data di dalam class tersebut biasa juga disebut dengan istilah atribut atau data member. Sedangkan operasinya disebut juga dengan istilah method atau member function. Class dibuat sesuai dengan kondisi nyata artinya bahwa atribut dan operasi di dalam sebuah class adalah memiliki hubungan yang sangat kuat dengan nama class-nya. Misalkan dibuat sebuah class bernama Mobil maka class tersebut setidaknya akan mempunyai atribut berupa jenis mobil, merk, isi bensin, kecepatan, dan tahun pembuatan. Sedangkan operasinya misalkan adalah berjalan, berhenti, tancap gas, injak rem dan nyalakan lampu.

Object sendiri merupakan instansiasi sebuah class. Class adalah template sedangkan object adalah realisasi dari template tersebut, sehingga ketika program dijalankan maka yang bertindak sebagai pelaku utamanya adalah object, bukan class. Bisa dianalogikan pada konsep procedural programming dengan class sebagai tipe data, sedangkan object adalah variabelnya (int angka;). Sebuah class bisa memiliki lebih dari satu object dengan syarat masing-masing object berbeda namanya. Misalnya class Mobil memiliki object Sedan, Pick_Up, Truk dan Bus (Mobil Sedan, Pick_Up, Truk, Bus;).

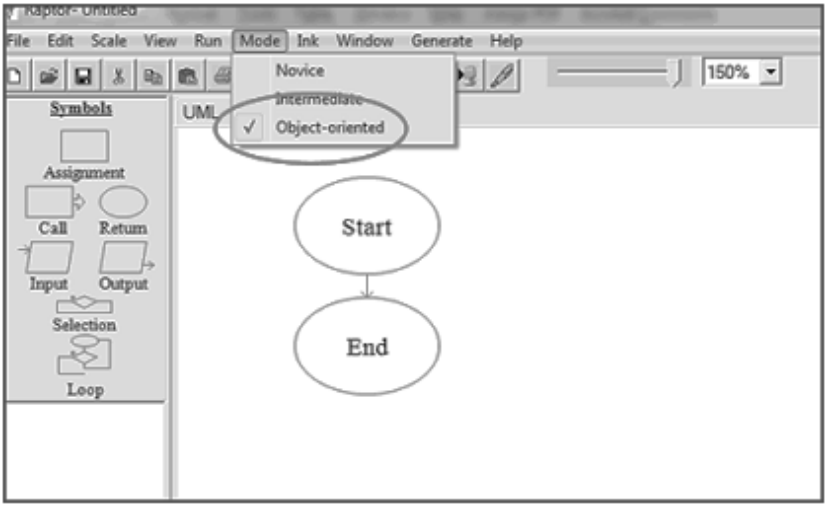
Di dalam OOP disediakan fasilitas data hiding (penyembunyian data). Fasilitas ini diimplementasikan dengan pemakaian visibility modifier di dalam sebuah class. Ada dua jenis visibility modifier dasar yaitu private dan public. Private artinya bagian class ini hanya bisa diakses oleh fungsi class yang bersangkutan, sedangkan public berarti bagian class ini bisa diakses oleh fungsinya sendiri maupun oleh

fungsi lain di luar class yang bersangkutan (misalnya dari class lain, atau dari fungsi utama). Umumnya bagian data member bersifat private, sedangkan bagian member function bersifat public.

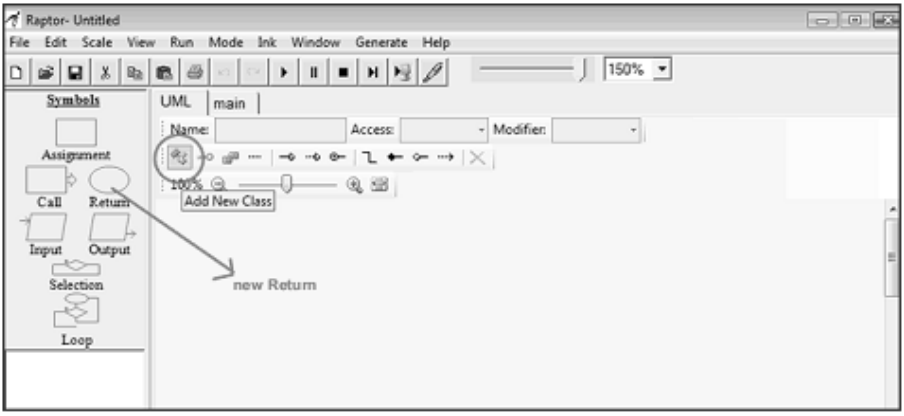
FLOWCHART UNTUK PEMROGRAMAN BERORIENTASI OBJEK

Pemrograman object-oriented mengharuskan untuk membuat class dimana terdapat method dan atribut didalamnya. RAPTOR tidak hanya dapat digunakan merancang program terstruktur tetapi juga pemrograman berorientasi objek (PBO).

Pilih **Mode → Object-oriented** untuk menggunakan RAPTOR dalam PBO.



Maka akan terdapat 2 tab, yaitu **UML** dan **main**. Tab **UML** digunakan untuk membuat struktur program berorientasi objek. **Class** dibuat pada tab UML, dengan cara klik tombol **add new class**. Simbol baru **Return** juga ditambahkan pada panel symbol.



Contoh:

Membuat flowchart PBO yang dapat melakukan perhitungan volume kubus.

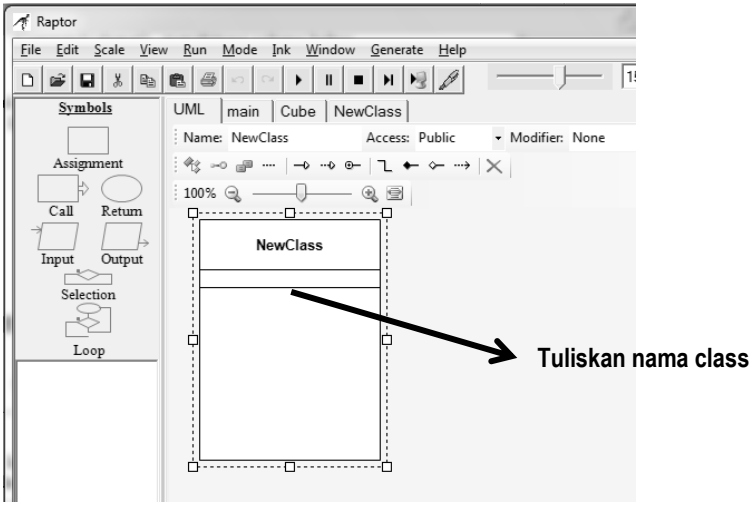
Komponen yang akan dibuat antara lain:

a. Nama class : Cube
b. Atribut : side dan volume
c. Method :
   - SetSide(), memberikan nilai besaran variable side (sisi kubus).
   - Getside(), membaca nilai variable side dalam memori.
   - ComputeVolume(), menghitung volume kubus.
   - GetVolume(), membaca hasil perhitungan ComputeVolume().

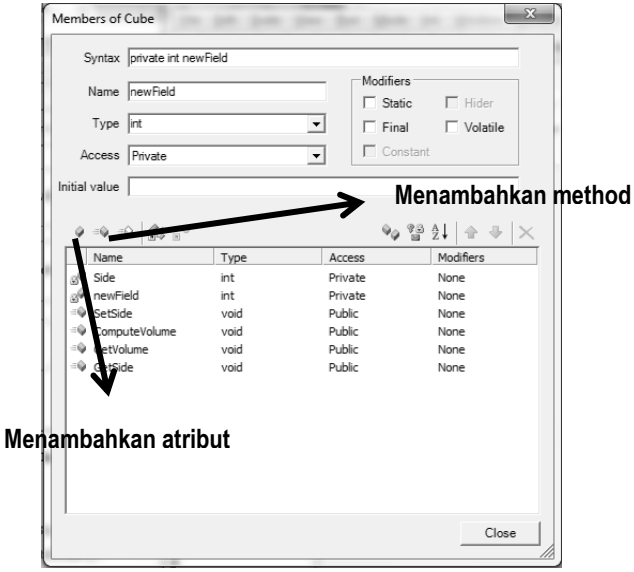
Langkah-langkah:

1. Klik tombol Add New Class pada tab UML.

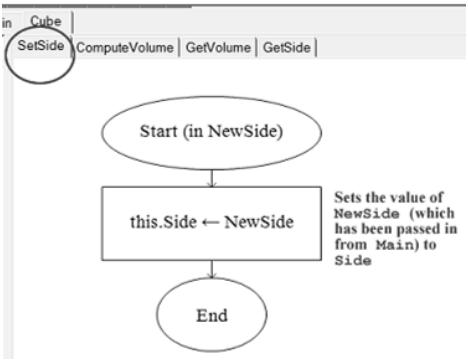
2. Klik symbol class dan beri nama class.

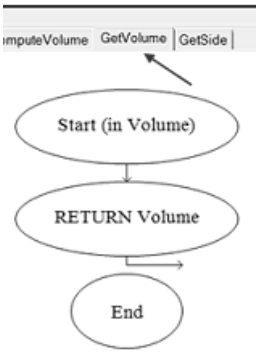


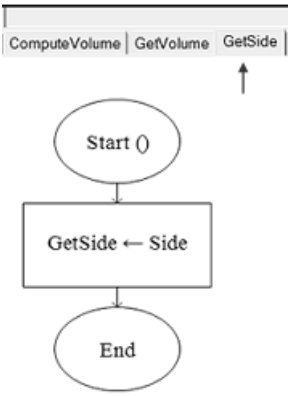
3. Klik ganda pada symbol class untuk menambahkan atribut dan metode.

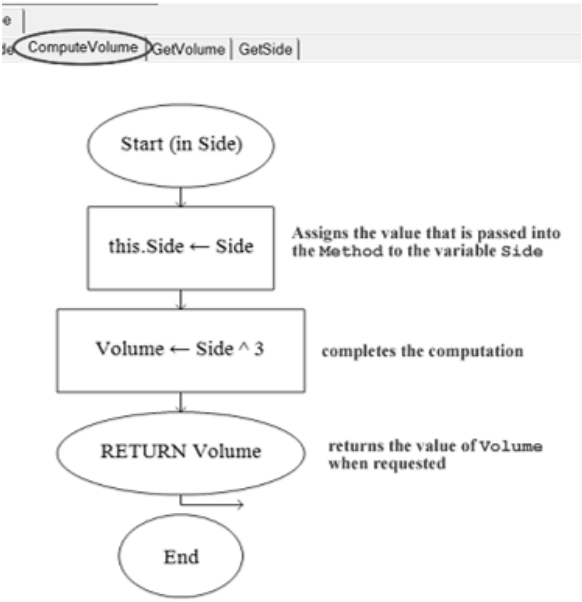


4. Klik tab class Cube, maka akan terdapat 4 tab method yang sudah dibuat pada langkah 3. Isi masing-masing method sesuai algoritma yang sudah dirancang (lihat gambar berikut).

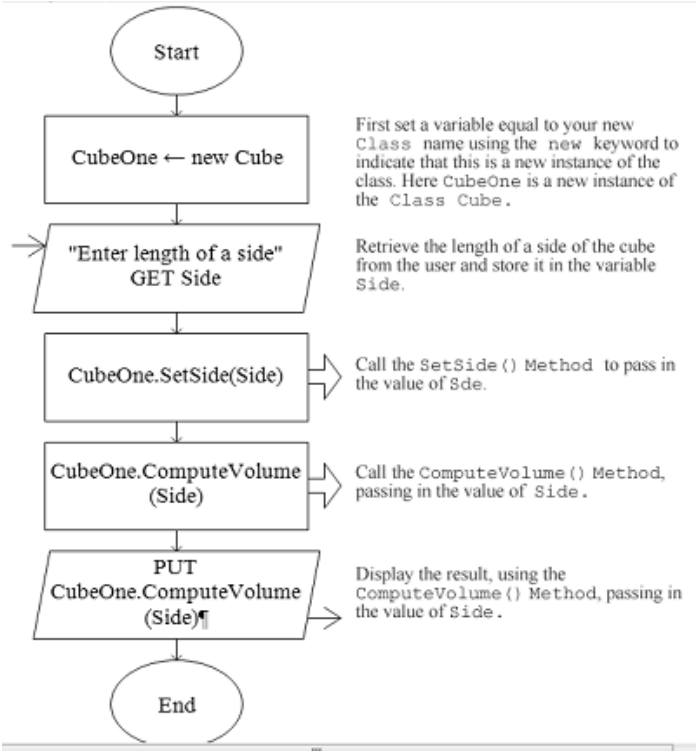






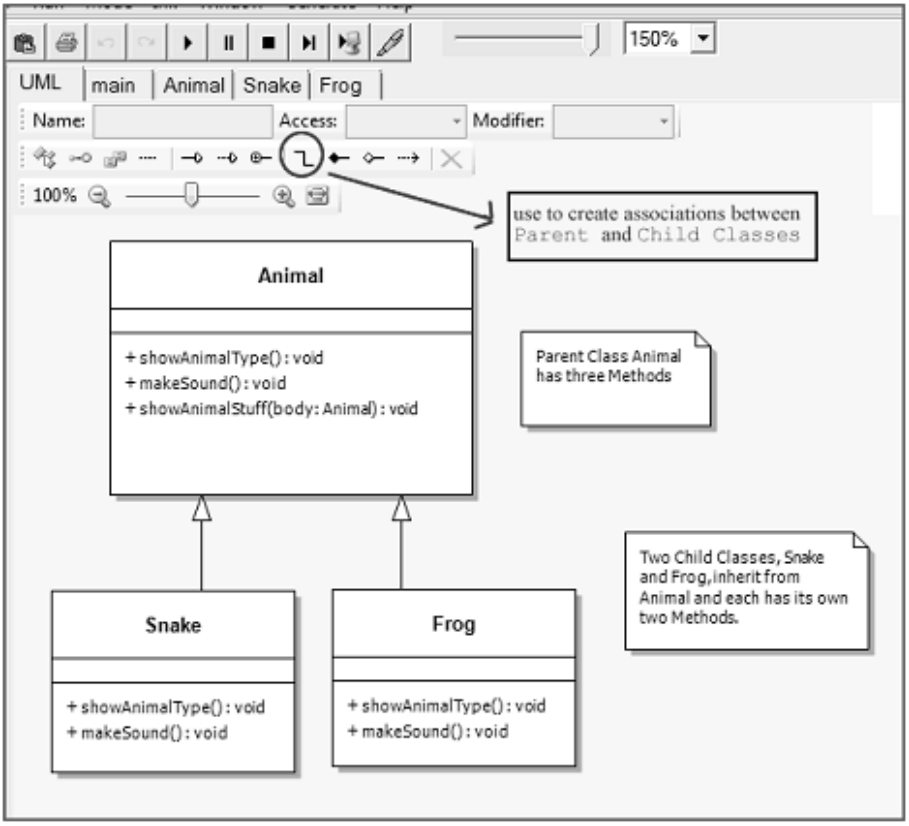


5. Buat algoritma pada tab main.



Inheritance

Dalam PBO memungkinkan terdapat pewarisan dari class parent ke class child. RAPTOR juga mengatasi hal ini dengan disediakannya tombol untuk membuat hubungan antara class parent dan class child. Lihat gambar berikut untuk ilustrasinya.



Untuk pembuatan flowchart menggunakan tools lain, seperti Ms. Visio atau fasilitas di Ms.Word, cukup mengadopsi saja aturan penggambaran flowchart OOP, yaitu:

1. Buat diagram UML (dapat diganti dengan shape kotak) yang mendefinisikan nama class, atribut class, dan method dalam class.
2. Flowchart dibuat untuk setiap method dalam class dan program yang memanggil class tersebut (program utama).
3. Atribut class tidak perlu didefinisikan di setiap method class, cukup atribut local dalam method saja yang didefinisikan dalam method tersebut.

LATIHAN DAN ANALISA :

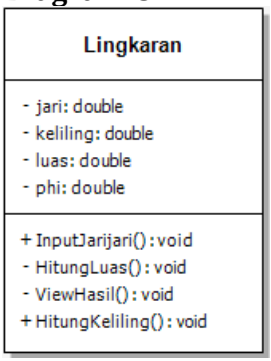
Membuat flowchart dan program tentang operasi geometri lingkaran.
Langkah-langkah :

1. Identifikasi class yang dibutuhkan.
   Dalam kasus ini class-nya diberi nama lingkaran.
2. Identifikasi atribut/ data member yang dimiliki oleh class.
   Sebuah lingkaran mempunyai atribut : jari-jari, luas dan keliling. Phi adalah konstanta, sehingga cukup dimasukkan langsung ke dalam perhitungan.
3. Identifikasi fungsi yang diperlukan oleh class, disesuaikan dengan data member yang dimiliki.
   Jika dilihat dari data member di atas maka dibutuhkan fungsi untuk :
   a. Memasukkan jari-jari,
   b. Menghitung luas (bukan memasukkan luas!),
   c. Menghitung keliling (bukan memasukkan keliling!), dan
   d. Menampilkan hasil perhitungan luas dan keliling.
4. Tentukan visibility modifier private atau public untuk setiap atribut dan fungsi di dalam class.
   Berdasar kondisi atribut dan operasi hasil analisa di atas maka dapat dimisalkan untuk semua data member (jari-jari, luas dan keliling) dimasukkan ke dalam private, artinya cukup diakses oleh fungsi internal class. Untuk member function, fungsi input jari-jari dan output luas-keliling dijadikan public sedangkan penghitungan luas dan keliling cukup dijadikan private. Karena fungsi penghitungan bersifat private maka harus dipastikan bahwa kedua fungsi tersebut dijalankan oleh fungsi yang bersifat public. Sebagai contoh untuk skenario ini akan dijalankan di dalam fungsi input dengan syarat dilakukan setelah pemasukan data jari-jari sudah dikerjakan.
5. Buat program utama sesuai dengan isi class.
   Skenario untuk program utamanya :

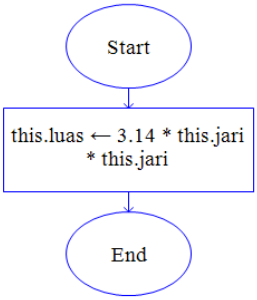
a. Dibuat obyek dari class lingkaran dengan nama lingkaran-ku.
b. Dipanggil fungsi input milik obyek lingkaran-ku.
c. Dipanggil fungsi output milik obyek lingkaran-ku.

CONTOH FLOWCHART

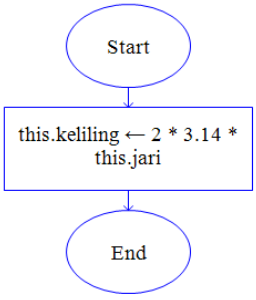
1. Diagram UML



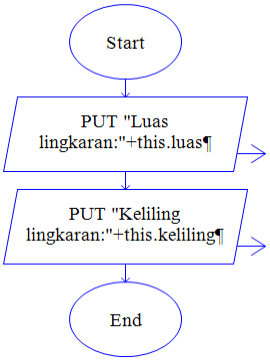
2. Flowchart setiap method



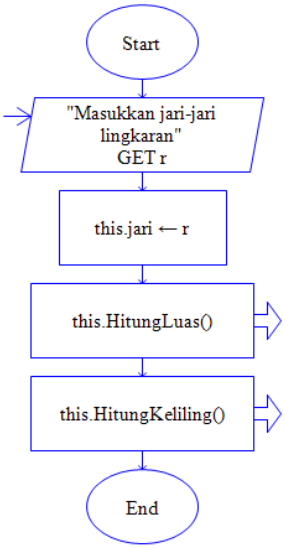
HitungLuas()



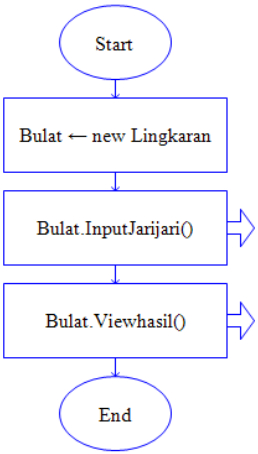
HitungKeliling()



ViewHasil()



Flowchart InputJarijari()



Flowchart Program Utama

CONTOH IMPLEMENTASI :

```
package samples;
import java.util.Scanner;

public class Lingkaran {
   double r,luas, keliling;
   public void HitungLuas(){
      this.luas=3.14*this.r*this.r;
   }
   public void HitungKeliling(){
      this.keliling=2*3.14*this.r;
   }
   public void InputJarijari(){
      System.out.println("Masukkan jari-jari:");
      Scanner userInput=new Scanner(System.in);
      this.r=userInput.nextDouble();
      HitungLuas();
      HitungKeliling();
   }
   public void ViewHasil(){
      System.out.println("Luas lingkaran:"+this.luas);
      System.out.println("Keliling lingkaran:"+this.keliling);
   }
```

```
    public static void main(String[] args){
       Lingkaran ling=new Lingkaran();
       System.out.println(":: PROGRAM LINGKARAN ::");
       ling.InputJarijari();
       ling.ViewHasil();
    }
}
```

SOAL :

1. Buatlah program OOP tentang universitas.
2. Buatlah program OOP tentang hotel
3. Buatlah program OOP tentang mahasiswa.
4. Buatlah program OOP tentang apotek.
5. Buatlah program OOP tentang barang di swalayan.

[Halaman ini sengaja dibiarkan kosong]

MODUL 2
CONSTRUCTOR DAN DESTRUCTOR

Proses inisialisasi data member di dalam sebuah class dilakukan oleh sebuah fungsi anggota (*member function*) yang disebut constructor. Fungsi ini bersifat unik karena mempunyai nama yang sudah ditentukan, yakni sama dengan nama class-nya. Fungsi constructor tidak memiliki *return type/return value* (nilai kembalian), namun bisa diberi argumen/parameter atau tidak. Constructor dijalankan secara otomatis ketika object dari sebuah class dibuat (umumnya di fungsi utama – main).

Destructor merupakan sebuah fungsi yang terdapat secara khusus didalam sebuah *class*. Destructor umumnya dipakai untuk menghapus object dari memori komputer supaya lokasi memori yang sebelumnya ditempatinya bisa dipakai oleh object yang lain. Pada bahasa pemrograman tertentu fungsi destructor perlu dituliskan dan akan dieksekusi secara otomatis pada akhir sebuah blok statement, namun dalam bahasa pemrograman *Java* fungsi destructor sudah tidak diperlukan lagi karena *Java* akan secara otomatis menghapus obyek atau variabel yang masuk kedalam *garbage collection*.

Dalam bahasa pemrograman *Java*, diijinkan terdapat lebih dari satu fungsi dengan nama yang sama asalkan memiliki parameter yang berbeda.

LATIHAN DAN ANALISA

Membuat program OOP tentang sepeda motor.
Langkah-langkah :

1. Identifikasi class yang diperlukan.
   Dalam kasus ini cukup ada sebuah class, misal diberi nama class motorbike.
2. Identifikasi atribut dan operasi dari class yang bersangkutan.
   Atributnya : merk, tangki bensin, plat nomor, kilometer tempuh.

Operasinya : jalan, isi bensin, berhenti.

3. Tentukan bagian public dan private.
   Bagian public : merk dan plat nomor serta ketiga operasi.
   Bagian private : tangki bensin dan kilometer tempuh. Diasumsikan cukup diketahui oleh pemilik sepeda motor.
4. Buat constructor yang sesuai.
   Dalam kasus ini disiapkan tiga skenario constructor :
   a. Inisialisasi data member tangki bensin dan kilometer tempuh dengan nilai 0 (nol) secara default constructor.
   b. Inisialisasi data member merk dengan nama "ANDALAN" secara parameterized constructor.
   c. Inisialisasi data member merk dan plat nomor dengan nama "ANDALAN" dan nomor plat "L XXXX BC" secara default argument pada data member merk.
5. Buat skenario fungsi utama.
   Di fungsi utama akan dijalankan :
   a. Pembuatan obyek class motorbike dengan nama sepeda_motor_1 untuk menjalankan constructor pertama.
   b. Pembuatan obyek class motorbike dengan nama sepeda_motor_2 untuk menjalankan constructor kedua.
   c. Pembuatan obyek class motorbike dengan nama sepeda_motor_3 untuk menjalankan constructor ketiga.

CONTOH IMPLEMENTASI :

```
package samples;

public class Motorbike {
   String platNomor;
   int tangkiBensin;
   int kmTempuh;
   public Motorbike(){
      this.tangkiBensin=0;
      this.kmTempuh=0;
      this.platNomor="KOSONG";
```

```
    }
    public Motorbike(int tb, int kt){
        this.tangkiBensin=tb;
        this.kmTempuh=kt;
        this.platNomor="KOSONG";
    }
    public Motorbike(int tb, int kt, String pn){
        this.tangkiBensin=tb;
        this.kmTempuh=kt;
        this.platNomor=pn;
    }
    public void tampilData(){
        System.out.println("Isi bensin: "+this.tangkiBensin);
        System.out.println("Kilometer: "+this.kmTempuh);
        System.out.println("Plat nomor: "+this.platNomor);
    }
    public static void main(String[] args){
        Motorbike s1=new Motorbike();
        Motorbike s2=new Motorbike(1,1);
        Motorbike s3=new Motorbike(1,1,"L4243AE");
        s1.tampilData();
        s2.tampilData();
        s3.tampilData();
    }

}
```

SOAL

1. Buatlah fungsi constructor untuk program OOP tentang Laptop.
2. Buatlah fungsi constructor untuk program OOP tentang Handphone.
3. Buatlah fungsi constructor untuk program OOP tentang Alat Tulis.
4. Buatlah fungsi constructor untuk program OOP tentang Hewan.
5. Buatlah fungsi constructor untuk program OOP tentang Karyawan.

Catatan : Lengkapi dengan contoh main programnya

[Halaman ini sengaja dibiarkan kosong]

MODUL 3
ARRAY OBJECT

Dalam pembuatan program OOP yang sebenarnya seringkali tidak cukup dengan adanya satu buah obyek saja akan tetapi dibutuhkan beberapa obyek yang diperoleh dari satu class yang sama. Untuk menangani hal itu maka dipakailah konsep array obyek, bukan dengan cara membuat obyek yang lebih dari satu. Array obyek berarti sebuah array (larik) yang setiap elemennya berisikan sebuah obyek dari class yang sama. Masing-masing obyek bisa diakses berdasarkan nomer indeks yang dimilikinya.

Penggunaan array obyek umumnya bertujuan untuk pengolahan data. Maksudnya adalah untuk operasi penambahan, pengeditan, pencarian dan penghapusan data.

LATIHAN DAN ANALISA

Membuat program OOP tentang mahasiswa.
Langkah-langkah :

1. Identifikasi class yang dibutuhkan.
   Dalam kasus ini cukup dibutuhkan sebuah class yang diberi nama mahasiswa.
2. Identifikasi atribut class.
   Data member : NPM, Nama, IPK.
   Member function : tambah data, tampilkan data dan cari data.
3. Penentuan bagian private dan public.
   Semua data member dimisalkan termasuk bagian private. Sedangkan semua member function dimisalkan termasuk bagian public.
4. Pembuatan constructor dan destructor jika perlu.
   Sementara ini tidak dibuat dulu.
5. Pembuatan skenario fungsi utama.
   a. Dibuat array obyek untuk class mahasiswa dengan nama mhs dengan jumlah elemen 5 buah.

b. Dijalankan fungsi tambah data melalui looping.
c. Dijalankan fungsi tampilkan data melalui looping.
d. Dijalankan fungsi cari data.

CONTOH IMPLEMENTASI

```
package samples;
import java.util.Scanner;

public class Mahasiswa {
   String NPM, nama;
   float ipk;
   public void tambah(String NPM, String nama, float ipk){
      this.NPM=NPM;
      this.nama=nama;
      this.ipk=ipk;
   }
   public void tampil(){
      System.out.println("NPM: "+this.NPM);
      System.out.println("Nama: "+this.nama);
      System.out.println("IPK: "+this.ipk);
   }
   public boolean cariNPM(String npmCari){
         if (this.NPM.equals(npmCari))
            return true;
         else return false;
   }
   public static void main(String[] args){
      Mahasiswa[] arrayMhs=new Mahasiswa[4];
      Mahasiswa mhs;
      String npm,nama;
      float ipk;
      System.out.println("Program Data Mahasiswa");
      System.out.println("=======================");
      for (int i=0;i<4;i++){
         System.out.println("Masukkan data NPM-"+i+":");
         Scanner userInput=new Scanner(System.in);
         npm=userInput.next();
         System.out.println("Masukkan data Nama-"+i+":");
         nama=userInput.next();
         System.out.println("Masukkan data IPK-"+i+":");
         ipk=userInput.nextFloat();
         mhs=new Mahasiswa();
         mhs.tambah(npm, nama, ipk);
```

```
            arrayMhs[i]=mhs;
        }
        for (int i=0;i<4;i++){
            arrayMhs[i].tampil();
            System.out.println("-----------------------------");
        }
        System.out.println("Pencarian Data:");
        System.out.println("Masukkan NPM yang dicari:");
        Scanner userInput=new Scanner(System.in);
        npm=userInput.next();
        Boolean hasil;
        int i=0;
        for (i=0;i<4;i++){
            hasil=arrayMhs[i].cariNPM(npm);
            if (hasil)
                break;
        }
        if (i==5)
            System.out.println("NPM tidak ditemukan");
        else
            System.out.println("NPM ditemukan");
    }
}
```

SOAL

1. Buatlah program OOP tentang pengolahan data Pegawai.
2. Buatlah program OOP tentang pengolahan data Calon Maba.
3. Buatlah program OOP tentang pengolahan data Pembelian tiket Pesawat.
4. Buatlah program OOP tentang pengolahan data Barang di swalayan.
5. Buatlah program OOP tentang pengolahan data Peminjaman VCD.

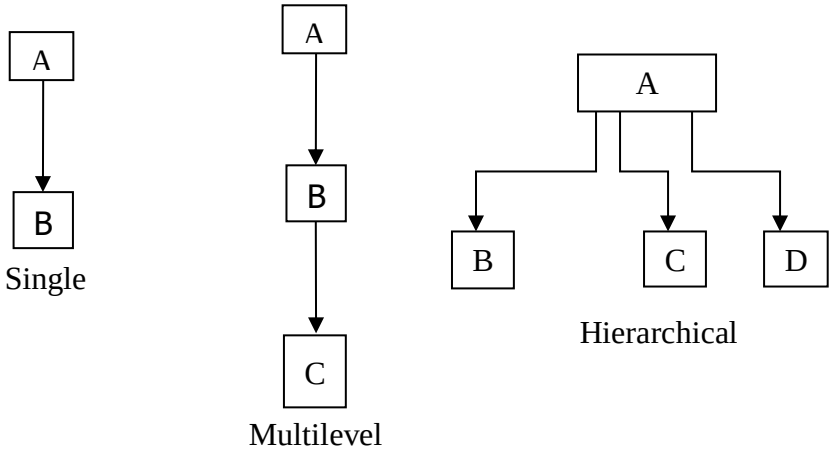
Catatan :
Lengkapi dengan EDIT dan HAPUS data serta program dibuat berupa pilihan menu.

[Halaman ini sengaja dibiarkan kosong]

MODUL 4
PEWARISAN (*INHERITANCE*)

Fitur lain yang disediakan oleh konsep OOP adalah adanya mekanisme inheritance/pewarisan. Ide dasar pewarisan adalah untuk meningkatkan kemampuan program supaya dapat dipakai kembali (reusability) ketika dibutuhkan. Dengan kata lain menghindari terjadinya redundancy (penulisan yang sama) source code. Dalam implementasinya sendiri, pewarisan berarti menurunkan beberapa atau bahkan semua atribut dari class induk ke class anak.

Struktur model pewarisan dalam bahasa pemrograman Java cukup sederhana seperti terlihat pada Gambar 4.1 berikut:



Gambar 4.1. Struktur Model Pewarisan

LATIHAN DAN ANALISA

Membuat program OOP tentang laporan nilai mahasiswa menggunakan konsep pewarisan.
Langkah-langkah :

1. Identifikasi class yang dibutuhkan, beserta jenis pewarisan yang sesuai.
   Dalam kasus laporan nilai mahasiswa ini dapat dianalisa berdasarkan proses diperolehnya nilai tersebut. Pertama kali mahasiswa akan mendapat nomor ujian. Kemudian dengan nomor ujian tersebut mahasiswa akan mengikuti tes untuk mendapatkan nilai. Dan selanjutnya akan dilakukan rekapitulasi nilai keseluruhan dengan mencantumkan nomor ujian mahasiswa beserta nilai-nilai yang diperolehnya. Berdasarkan proses ini maka jenis pewarisan yang sesuai adalah multilevel.
2. Identifikasi atribut dan fungsi class yang ada.
   Class mahasiswa :
   - Atribut : nomor ujian.
   - Fungsi : mendapat nomor ujian dan menampilkan nomor ujian.

   Class ujian :
   - Atribut : nilai 1 dan nilai 2.
   - Fungsi : mengisi nilai dan menampilkan nilai.

   Class laporan :
   - Atribut : --
   - Fungsi : menampilkan jumlah nilai. Dalam fungsi tersebut juga dijalankan fungsi untuk menampilkan nomor ujian serta nilai-nilai yang didapat.
3. Pembuatan constructor jika perlu.
   Sementara tidak diperlukan.
4. Pembuatan skenario fungsi utama.
   a. Membuat obyek cukup untuk class laporan dengan nama raport. Class yang lain belum perlu dibuatkan obyeknya karena semua atribut milik class induk telah diwariskan ke anaknya, tidak ada bagian private di dalam class induk. Pewarisan multilevel secara public mengakibatkan semua bagian protected dan bagian public class mahasiswa diwariskan ke class ujian. Kemudian pewarisan dari class ujian ke class laporan

secara public juga mengakibatkan semua bagian protected dan public class ujian diwariskan ke class laporan.

b. Menjalankan fungsi isi nomor laporan.
c. Menjalankan fungsi isi nilai-nilai ujian.
d. Menjalankan fungsi rekap jumlah nilai.

CONTOH IMPLEMENTASI

```
package another;
public class Mahasiswa {
   String noUjian;
   int nilai1, nilai2;
   public void setNoUjian(String no){
      this.noUjian=no;
   }
   public void cetakNoUjian(){
      System.out.println("Nomor = "+this.noUjian);
   }
   public static void main(String[] args){
      Laporan rapor = new Laporan();
      rapor.setNoUjian("123");
      rapor.setMarks(85, 86);
      rapor.display();
   }
}

class Ujian extends Mahasiswa {
   public void setMarks(int x, int y){
      this.nilai1=x;
      this.nilai2=y;
   }
   public void cetakMarks(){
      System.out.println("Nilai 1 = "+this.nilai1);
      System.out.println("Nilai 2 = "+this.nilai2);
   }
}
class Laporan extends Ujian {
   public void display(){
      cetakNoUjian();
      cetakMarks();
      System.out.println("Total = "+(this.nilai1+this.nilai2));
   }
}
```

SOAL

Buatlah program OOP lengkap dengan pewarisan minimal melibatkan 3 class tentang:

a. Penjualan voucher pulsa HP.
b. Penghitungan nilai akhir kuliah.
c. Pendataan koleksi hewan kebun binatang.
d. Penghitungan gaji pegawai.
e. Peminjaman buku perpustakaan.

MODUL 5
OVERLOADING FUNGSI

Di dalam konsep OOP terdapat fasilitas untuk melakukan overload terhadap sebuah fungsi (baik fungsi lepas maupun fungsi anggota sebuah class). Didalam bahasa pemrograman Java, overload fungsi berarti membuat sebuah fungsi mampu melakukan berbagai tugas yang berbeda. Perbedaan tugas fungsi ditentukan oleh banyaknya parameter yang dimiliki sebuah fungsi yang dilakukan overload. Dalam bahasa pemrograman Java, overloading fungsi dapat dilakukan dengan menggunakan parameter yang berbeda, baik dari segi jumlah maupun tipe, dengan fungsi yang memiliki nama yang sama. Sebagai contoh sederhananya adalah berikut ini :

```
public class BangunDatar {
  String nama;
  int radius,panjang,lebar;
  public BangunDatar(String n){
    nama=n;
  }
  public BangunDatar(String n,int r){
    nama=n;
    radius=r;
  }
  public BangunDatar(String n, int p, int l){
    nama=n;
    panjang=p;
    lebar=l;
  }
}
```

Hasil overload fungsi secara umum dapat dilihat dari bentuk pemanggilan fungsinya. Ada fungsi yang dipanggil tanpa parameter, ada yang dipanggil dengan satu buah parameter atau mungkin ada fungsi yang dipanggil dengan beberapa parameter sekaligus. Semua itu tergantung pada kebutuhan programnya.

LATIHAN DAN ANALISA:

Membuat program tentang operasi penjumlahan pada sebuah bilangan.
Langkah-langkah :

1. Identifikasi class yang dibutuhkan.
   Dalam kasus ini dibutuhkan satu buah class. Misal diberi nama class angka.
2. Identifikasi atribut dari class yang bersangkutan.
   Atribut : bilangan bulat 1, bilangan bulat 2, bilangan pecahan 1 dan bilangan pecahan 2.
   Operasi : penjumlahan yang di-overload sesuai kondisi bilangan. Ada penjumlahan sesama bilangan bulat, ada yang campuran dan ada yang sesama bilangan pecahan.
3. Penentuan bagian private dan public.
   Semua atribut diletakkan di bagian private, sedangkan operasi diletakkan di bagian public.
4. Pembuatan constructor dan destructor jika perlu.
   Sementara ini tidak diperlukan.
5. Pembuatan skenario fungsi utama.
   a. Dibuat obyek dari class angka dengan nama bilangan.
   b. Dijalankan semua jenis fungsi overload satu per satu sesuai kondisinya.

CONTOH IMPLEMENTASI

```
package samples;

public class Angka {
   int bulat1,bulat2;
   double p1,p2;
   public void tambah(){
      this.bulat1=10;
      this.bulat2=20;
      System.out.println("Hasil= "+(this.bulat1+this.bulat2));
   }
   public void tambah(int b1, int b2){
      this.bulat1=b1;
      this.bulat2=b2;
      System.out.println("Hasil= "+(this.bulat1+this.bulat2));
```

```
    }
    public void tambah(double f1, double f2){
        this.p1=f1;
        this.p2=f2;
        System.out.println("Hasil="+(this.p1+this.p2));
    }
    public static void main(String[] args){
        Angka a=new Angka();
        a.tambah();
        a.tambah(2,3);
        a.tambah(1.2, 2.5);
    }
}
```

SOAL

1. Analisa dan buatlah overload fungsi untuk program OOP tentang pembelian buku.
2. Analisa dan buatlah overload fungsi untuk program OOP tentang penggajian pegawai.
3. Analisa dan buatlah overload fungsi untuk program OOP tentang pembayaran daftar ulang mahasiswa baru.
4. Analisa dan buatlah overload fungsi untuk program OOP tentang penjualan tiket pesawat.
5. Analisa dan buatlah overload fungsi untuk program OOP tentang pencatatan sensus penduduk.

Catatan : Lengkapi dengan contoh fungsi utamanya.

[Halaman ini sengaja dibiarkan kosong]

MODUL 6
VIRTUAL FUNCTION

Konsep polymorphism di modul awal sudah diimplementasikan dalam bentuk overloading fungsi maupun operator. Dalam modul ini akan diimplementasikan fitur polymorphism dalam paradigma OOP menggunakan virtual function. Polymorphism jenis ini dikenal dengan istilah running time polymorphism. Sedangkan overloading dikenal dengan istilah compile time polymorphism.

Virtual function seringkali dipakai dalam program OOP yang menggunakan fitur pewarisan. Misalnya terjadinya nama fungsi yang sama antara class induk dengan class anak. Manakah yang akan diakses ketika digunakan sebuah pointer yang sama dari hasil class turunan? Dalam bahasa pemrograman Java, semua fungsi merupakan virtual function, sehingga secara otomatis jika dideklarasikan sebuah fungsi dengan nama yang sama maka akan dilakukan override terhadap fungsi pada class induk.

Untuk lebih jelasnya bisa dilihat pada contoh sederhana berikut ini :

```
public class Base {
 public void display(){
    System.out.println("Display base...");
  }
  public static void main(String[] args){
    Derived d=new Derived();
    d.display();
  }
}

class Derived extends Base{
 public void display(){
    System.out.println("Display derived...");
  }
}
```

Pada contoh di atas dapat dilihat bahwa kedua class sama-sama mempunyai dua buah fungsi yang sama, yaitu `display`. Kemudian dijalankan fungsi display. Fungsi display akan menjalankan display yang dimiliki oleh class turunan disebabkan adanya deklarasi virtual untuk fungsi display karena seluruh fungsi pada bahasa pemrograman Java merupakan virtual.

LATIHAN DAN ANALISA

Membuat program OOP yang menerapkan fungsi virtual tentang koleksi toko.
Langkah-langkah :

1. Identifikasi class-class yang dibutuhkan, beserta dengan model inheritance-nya.
   Misal ada tiga class yaitu media, buku dan kaset. Model pewarisannya adalah hirarki dengan class media sebagai class induk, sedangkan buku dan kaset sebagai class turunannya. Pewarisan bersifat public semua.
2. Identifikasi atribut dan fungsi masing-masing class.
   Class media berisi atribut-atribut umum yang akan diwariskan ke kedua turunannya. Isinya adalah judul dan harga. Fungsinya terdiri dari sebuah constructor dan fungsi untuk menampilkan data member tanpa ada isinya (karena cuma virtual).
   Class buku berisi data jumlah halaman. Sedangkan fungsinya terdiri dari sebuah constructor dan fungsi untuk menampilkan data judul, harga dan jumlah halaman.
   Class kaset berisi data durasi. Sedangkan fungsinya terdiri dari sebuah constructor dan fungsi untuk menampilkan data judul, harga dan durasi kaset.
3. Menentukan bagian private dan public.
   Class media : data member bersifat protected. Fungsi bersifat public.
   Class buku : jumlah halaman bersifat private. Fungsi bersifat public.

Class kaset : durasi bersifat private. Fungsi bersifat public.

4. Membuat constructor dan destructor.
   Constructor dibuat baik di class media, buku maupun kaset. Perlu diperhatikan untuk jenis contructor pada class turunan.
5. Membuat skenario fungsi utamanya.
   a. Membuat obyek untuk class buku dengan nama bukuku.
   b. Membuat obyek untuk class kaset dengan nama kasetku.
   c. Membuat obyek untuk class media dengan nama mediaku berupa array bertipe pointer obyek sebanyak dua elemen.
   d. Menyimpan alamat obyek buku di mediaku pada elemen ke-0 dan menyimpan alamat obyek kaset di mediaku pada elemen ke-1.
   e. Menampilkan isi obyek buku dan obyek kaset melalui akses lewat pointer ke virtual function.

CONTOH IMPLEMENTASI

```
package samples;
import java.util.ArrayList;

public class Media {
   String judul;
   int harga;
   public Media(String s, int h){
      this.judul=s;
      this.harga=h;
   }
   public void display(){
   }
   public static void main(String[] args){
      Buku b=new Buku("Object Oriented Programming",55000,224);
      Kaset k=new Kaset("Belajar Dasar-dasar OOP",15000,2);
      ArrayList<Media> mediaku=new ArrayList<Media>(2);
      mediaku.add(b);
      mediaku.add(k);
      System.out.println("Daftar Barang:");
      mediaku.get(0).display();
      mediaku.get(1).display();
```

```
   }
}

class Buku extends Media {
   int halaman;
   public Buku(String s, int h, int hal){
      super(s,h);
      this.halaman=hal;
   }
   public void display(){
      System.out.println("Judul: "+this.judul);
      System.out.println("Halaman: "+this.halaman);
      System.out.println("Harga: "+this.harga);
   }
}

class Kaset extends Media {
   int durasi;
   public Kaset(String s, int h, int d){
      super(s,h);
      this.durasi=d;
   }
   public void display(){
      System.out.println("Judul: "+this.judul);
      System.out.println("Durasi: "+this.durasi);
      System.out.println("Harga: "+this.harga);
   }
}
```

SOAL

Implementasikan virtual function ke dalam program OOP tentang :

1. Kasir Swalayan
2. Teller Bank
3. Gudang Suku Cadang Pesawat
4. Servis Mobil Pribadi
5. Penjualan Rumah di Perumahan

Catatan :

- Untuk implementasi virtual function dibutuhkan pewarisan class terlebih dahulu.
- Akses virtual function harus melalui pointer ke obyek.

MODUL 7
ABSTRACT CLASS DAN INTERFACE JAVA

1. Abstract Class dan Abstract Method

Dalam sebuah program Java, kita dapat membuat sebuah class yang fungsinya hanya untuk memberikan karakteristik fundamental (dasar) untuk class-class yang lain. Class yang seperti ini tidak dapat digunakan untuk mendeklarasikan variabel (atau dengan kata lain, dibuat objeknya) dan disebut sebagai class abstrak (*abstract*). Class yang dideklarasikan abstrak hanya dijadikan sebagai *parent class* dari class turunannya.

Semua subclass yang mengekstensi dari superclass abstrak harus mengimplementasikan semua method abstrak yang ada dalam superclass. Berikut adalah contoh class abstrak `MyClass`:

```
package samples;

public abstract class MyClass {
   //data members
   private int myDataMember;

   public MyClass(int md){
      //method konkrit memiliki definisi yang jelas
      myDataMember=md;
   }
   public int getData(){
      //method konkrit lain
      return myDataMember;
   }
   public abstract int calc(int factor);
   //method abstrak tidak memiliki definisi
   }
```

Berikut contoh class `AnotherClass` yang menggunakan class abstrak `MyClass` di atas:

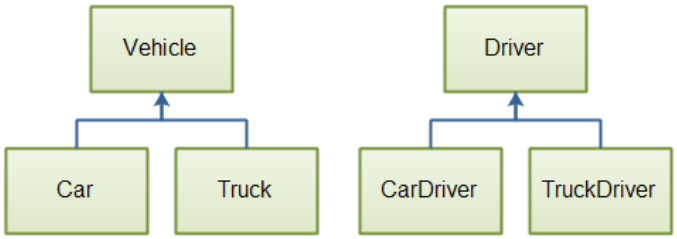
```
package samples;

public class AnotherClass extends MyClass{
  public AnotherClass (int md){
     //memanggil konstruktor dari superclass
     super(md);
  }
  //mengimplementasikan semua method abstrak
  public int calc(int factor){
     return factor*factor;
  }
}
```

Class `AnotherClass` harus mengimplementasikan method abstrak `calc(int factor)` yang dideklarasikan di `MyClass`.

2. Interface

Java memiliki konsep yang bernama *interface*. Interface Java mirip dengan class, tetapi hanya memiliki method dan konstanta saja. Di dalam interface tidak ada variabel/atribut dan interface tidak dapat diinstansiasi seperti layaknya class biasa.

Interface digunakan untuk menambahkan sejumlah fitur terhadap sebuah class, tanpa harus memasukkan fitur tersebut di dalam class.



Gambar 7.1. Contoh Diagram UML Sederhana (Sumber: http://tutorials.jenkov.com/java/interfaces.html)

Gambar 7.1 menunjukkan diagram UML dari class yang digunakan dalam sebuah program, dimana class `Car` dan `Truck` adalah turunan dari class `Vehicle` sedangkan class `CarDriver` dan `TruckDriver` adalah turunan dari class `Driver`. Jika kita ingin menyimpan objek-objek dari class tersebut ke database, maka objek harus melalui proses serialisasi-deserialisasi. Secara singkat, proses serialisasi mengkonversi objek menjadi stream biner yang kemudian dapat disimpan dalam database dengan tipe data blob (*binary large object*). Untuk membaca kembali objek dari database ke dalam program Java, maka perlu dilakukan proses deserialisasi.

Misal proses serialisasi-deserialisasi kita masukkan ke dalam sebuah method `storeToDatabase()` dan diletakkan pada class `Storable`. Agar dapat diakses oleh semua objek dari class `Vehicle` dan `Driver`, kita buat `Storable` menjadi superclass di atas `Vehicle` dan `Driver` sehingga method `storeToDatabase()` dapat digunakan oleh class di bawahnya. Akan tetapi, hal ini akan mengakibatkan hirarki class menjadi kacau secara konseptual karena diagram tidak lagi memodelkan class vehicle dan driver saja, tetapi juga terikat pada mekanisme penyimpanan dan proses serialisasi.

Salah satu solusinya adalah dengan membuat interface yang memiliki deklarasi method `storeToDatabase()`. Class yang perlu untuk menyimpan objek ke database dapat mengimplementasikan interface tersebut dan mengimplementasikan method `storeToDatabase()`. Contohnya:

```
public interface Storable {
    public void storeToDatabase();
}
```

Lalu untuk menggunakannya, kita dapat melakukan seperti berikut:

```
public class Vehicle implements Storable{
    public void storeToDatabase(){
       //sejumlah langkah untuk penyimpanan ke
       //database
    public static void main()(String[] args){
       Vehicle v = new Vehicle();
       v.storeToDatabase();
    }
}
```

Dengan menggunakan interface, penulisan program yang membutuhkan fungsionalitas tambahan menjadi lebih bersih dan rapi ketimbang menggunakan konsep pewarisan/inheritance.

LATIHAN DAN ANALISA

1. Membuat program OOP dengan menerapkan konsep abstract class. Perhatikan code berikut ini:

```
abstract class Employee {
  private String name, address;
  protected int basicSalary;

  public String getName(){
     return name;
  }

  public String getAddress(){
     return address;
  }

  public int getBasicSalary(){
     return basicSalary;
  }

  public void setAddress(String add){
     address = add;
  }

  public void setName(String nm){
     name = nm;
  }

  public void setBasicSalary(int  sal){
     basicSalary = sal;
  }
  public abstract int getMonthlySalary();

}
```

Langkah-langkah:

1) Tulis ulang class `Employee` di atas dalam sebuah program Java dan beri nama sesuai nama class-nya (`Employee.java`).
2) Buat sebuah class lain yang bernama `NormalEmployee` yang merupakan turunan dari class `Employee`. Class ini harus memiliki satu method yang menghitung gaji bulanan (*monthly salary*) untuk seorang pegawai. Untuk pegawai biasa (*normal employee*), gaji bulanannya adalah sama dengan gaji pokoknya (*basic salary*).
3) Buat sebuah class lain yang bernama `BonusEmployee` yang merupakan turunan dari class `Employee`. Class ini mendefinisikan seorang pegawai yang mendapatkan tambahan bonus bulanan (*monthly bonus*) selain gaji pokoknya setiap bulan. Catatan: Anda tambahkan sendiri atribut yang diperlukan dalam class ini, berikut method setter dan getter-nya.

2. Membuat program OOP dengan menggunakan interface. Langkah-langkahnya sebagai berikut:
   1) Buat sebuah interface dengan nama `Operations` dan simpan dengan nama `Operations.java`. Interface ini mendefinisikan dua method:
      a. `insert(int a)`: menambahkan sebuah integer `a` ke dalam sekumpulan integer lainnya dalam sebuah array. Method ini tidak memiliki return value dan menerima 1 parameter `a` bertipe `int`.
      b. `isIn(int a)`: mengecek apakah sebuah integer `a` ada dalam kumpulan integer dalam array atau tidak. Method ini bersifat public, memiliki return value bertipe `Boolean`, dan menerima 1 parameter `a` bertipe `int`.
   2) Tulis program untuk sebuah class dengan nama `IntCollection` yang mengimplementasikan interface yang sudah Anda buat pada langkah di atas, simpan dengan nama `IntCollection.java`. Kerangka dari class `IntCollection` adalah sebagai berikut:

```
public class IntCollection implements Operations{
   private int size, noOfItems;
   private int[] intArray;

   public IntCollection(int size){
        // Code untuk constructor dengan 1 parameter
        // size
   }

   public IntCollection(){
        // Code untuk constructor tanpa parameter,
        // asumsi array berukuran 100

   }

   public int getNoOfItems(){
        // Code untuk getNoOfItems, mengembalikan
        // jumlah item yang ada dalam array
   }

   public boolean isIn(int val){
        // Code untuk method isIn
   }

   public void insert(int val){
        // Code untuk method insert
   }
}
```

3) Masukkan method main berikut ke dalam class IntCollection:

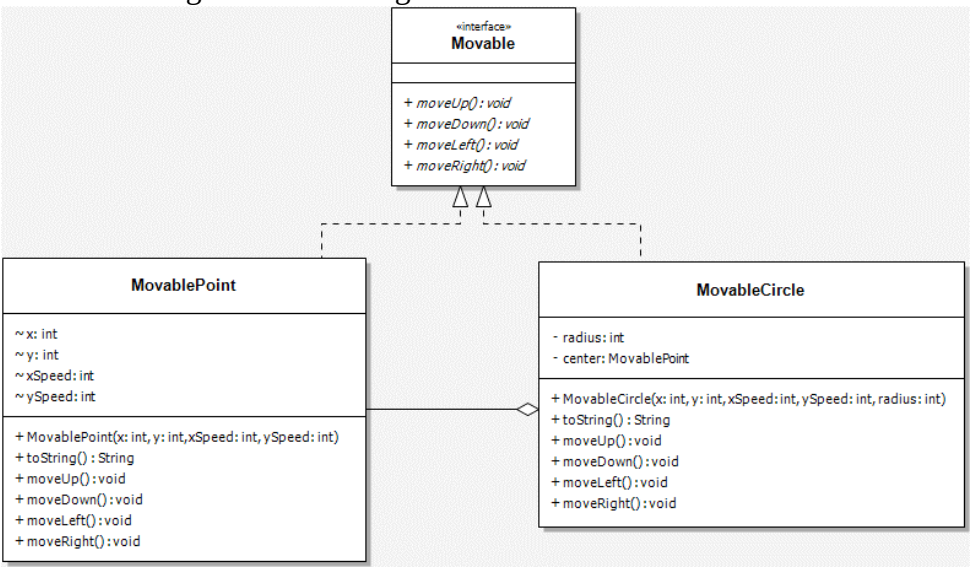
```
public static void main (String[] args){

   IntCollection in = new IntCollection(50);
   in.insert(99);
   in.insert(1);
   in.insert(2);
   in.insert(9);
   in.insert(44);
   in.insert(200);
   if(in.isIn(44)) System.out.println("Ok 1");
   if(in.isIn(99))
   System.out.println("Ok 2");
```

```
    if(in.isIn(200)) System.out.println("Ok 3");
    if(in.isIn(9)) System.out.println("Ok 4");
    if(!in.isIn(33))System.out.println("Ok 5");
    if(!in.isIn(441))System.out.println("Ok 6");
  }
```

4) Tuliskan output dari program di atas

SOAL

Perhatikan diagram UML sebagai berikut:



Buat program Java dari class-class di atas dengan mengikuti langkah-langkah sebagai berikut:

1. Untuk class `MovablePoint`, deklarasikan variabel dengan tanda tilde (~) sebagai variabel yang dapat diakses secara langsung oleh class lain dalam package yang sama.
2. Untuk class `MovableCircle`, gunakan sebuah `MovablePoint` sebagai titik tengahnya. Dengan kata lain, sebuah `MovableCircle` terdiri dari sebuah `MovablePoint` dan nilai radiusnya.

3. Untuk masing-masing class, buat konstruktor yang menginisialisasi tiap variabelnya.
4. Untuk masing-masing class, implementasikan tiap method yang berasal dari interface yang digunakan dan method milik class itu sendiri. Untuk bidang 2 dimensi yang digunakan, asumsikan posisi (0,0) adalah di pojok kiri atas. Ubah tiap nilai pada sumbu x dan y sesuai dengan arah pergerakannya.
5. Tulis program pada bagian main untuk mengetes class yang telah dibuat dan gunakan perintah berikut ini:

```
Movable m1 = new MovablePoint(5,6,10);
System.out.println(m1);
m1.moveLeft();
System.out.println(m1);

Movable m2 = new MovableCircle(2,1,2,20);
System.out.println(m2);
m2.moveRight();
System.out.println(m2);
```

Bagaimana output dari program di atas? Tuliskan kesimpulan Anda tentang penggunaan abstract class dan interface dalam laporan.