**LAPORAN RESMI**

Praktikum 8 Enkapsulasi

Mata Kuliah: Praktek Pemrograman Berbasis Objek



Disusun oleh:

M. Ainur Ramadhan (3122500047)

2 D3 Teknik Informatika B

Dosen Pengampu: Yanuar Risah Prayogi S.Kom., M.Kom.

**PROGRAM STUDI D3 TEKNIK INFORMATIKA**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2023/2024**

1. TUGAS PENDAHULUAN
   1. Apakah yang dimaksud dengan enkapsulasi?

Jawab:

Enkapsulasi adalah konsep dalam pemrograman berorientasi objek yang menggabungkan data dan metode yang beroperasi pada data ke dalam satu objek, dan umumnya melindungi data dari akses langsung dari luar objek. Ini memungkinkan pengendalian akses yang ketat terhadap data, abstraksi detail implementasi, dan memungkinkan perubahan internal yang aman. Dengan enkapsulasi, program dapat dirancang dengan lebih terstruktur, modular, dan dapat di-maintain, serta membantu dalam meningkatkan keamanan data.

* 1. Apakah yang dimaksud dengan constructor?

Jawab:

Constructor adalah metode khusus dalam pemrograman berorientasi objek (OOP) yang digunakan untuk menginisialisasi objek saat objek tersebut dibuat atau diinstansiasi dari sebuah kelas. Constructor memiliki nama yang sama dengan kelas tempatnya berada, dan mereka tidak mengembalikan nilai (tidak memiliki tipe kembalian).

* 1. Apakah yang dimaksud dengan overloading constructor?

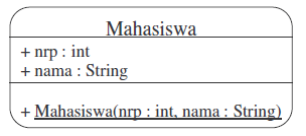
Jawab:

Overloading constructor (konstruktor overload) adalah konsep dalam pemrograman berorientasi objek di mana sebuah kelas dapat memiliki beberapa constructor dengan nama yang sama tetapi dengan daftar parameter yang berbeda. Dalam kata lain, kita dapat membuat beberapa versi constructor dalam kelas yang dapat menerima berbagai jenis dan jumlah argumen.

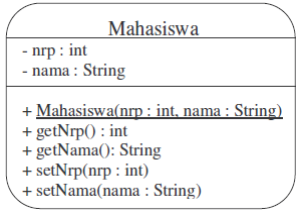
1. PERCOBAAN

Percobaan 1 : Melakukan enkapsulasi pada suatu class

Implementasikan UML class diagram Mahasiswa sebelum dan setelah dilakukan enkapsulasi!



Jika enkapsulasi dilakukan pada class diagram diatas, maka akan berubah menjadi:



Berikut implementasi kode program java untuk UML tersebut

* Sebelum diEnkapsulasi

public class Mahasiswa {

    private int nrp;

    private String nama;

    Mahasiswa(int nrp, String nama){

        this.nrp = nrp;

        this.nama = nama;

    }

}

* Sesudah diEnkapsulasi

public class Mahasiswa {

    private int nrp;

    private String nama;

    Mahasiswa(int nrp, String nama){

        this.nrp = nrp;

        this.nama = nama;

    }

    public int getNrp() {

        return nrp;

    }

    public String getNama() {

        return nama;

    }

    public void setNrp(int nrp) {

        this.nrp = nrp;

    }

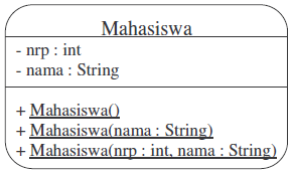
    public void setNama(String nama) {

        this.nama = nama;

    }

}

Percobaan 2 : Melakukan overloading constructor



Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

public class Mahasiswa {

    public int nrp;

    public String nama;

    public Mahasiswa(){

        nrp= 0;

        nama= "";

    }

    public Mahasiswa(String nama){

        nrp= 0;

        this.nama = nama;

    }

    public Mahasiswa(int nrp, String nama){

        this.nrp= nrp;

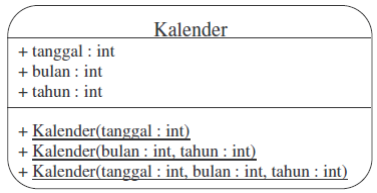
        this.nama= nama;

    }

}

1. LATIHAN

Latihan 1 : Mengimplementasikan UML class diagram dalam program untuk class Kalender



Dari class diagram diatas, desainlah suatu class yang memenuhi konsep enkapsulasi.

Untuk nilai inisialisasi, dipakai 1-1-2000. Pakailah kata kunci this untuk mempersingkat

pengkodean. Tulislah listing program berikut ini sebagai pengetesan

public class TesKalender {

    public static String getTime(Kalender kal) {

        String tmp;

        tmp = kal.getTanggal() + "-" +

                kal.getBulan() + "-" +

                kal.getTahun();

        return tmp;

    }

    public static void main(String args[]) {

        Kalender kal = new Kalender(8,1,2000);

        System.out.println("Waktu awal : " + getTime(kal));

        kal.setTanggal(9);

        System.out.println("1 hari setelah waktu awal : " + getTime(kal));

        kal = new Kalender(1,6,2003);

        System.out.println("Waktu berubah : " + getTime(kal));

        kal.setBulan(7);

        System.out.println("1 bulan setelah itu : " + getTime(kal));

        kal = new Kalender(20, 10, 2004);

        System.out.println("Waktu berubah : " + getTime(kal));

        kal.setTahun(2005);

        System.out.println("1 tahun setelah itu : " + getTime(kal));

    }

}

Berikut Kode Program untuk mengimplementasikan class Diagram Kalender

* *Kalender.java*

public class Kalender {

    private int tanggal;

    private int bulan;

    private int tahun;

    public Kalender(int tanggal){

        bulan= 0;

        tahun= 0;

        this.tanggal= tanggal;

    }

    public Kalender(int bulan, int tahun){

        tanggal=0;

        this.bulan = bulan;

        this.tahun = tahun;

    }

    public Kalender(int tanggal, int bulan, int tahun){

        this.tanggal= tanggal;

        this.bulan= bulan;

        this.tahun= tahun;

    }

    public int getTanggal() {

        return tanggal;

    }

    public int getBulan() {

        return bulan;

    }

    public int getTahun() {

        return tahun;

    }

    public void setTanggal(int tanggal) {

        this.tanggal = tanggal;

    }

    public void setTahun(int tahun) {

        this.tahun = tahun;

    }

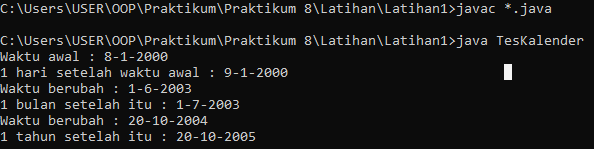
    public void setBulan(int bulan) {

        this.bulan = bulan;

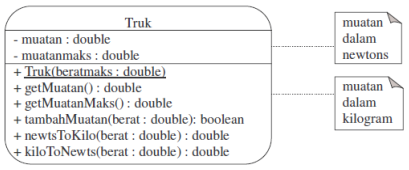
    }

}

Output:



Latihan 2 : Mengimplementasikan UML class diagram dalam program untuk class Truk



Keterangan : 1 kilogram = 9,8 newtons

Tranformasikan class diagram diatas ke dalam bentuk program! Tuliskan listing program berikut ini sebagai pengetesan.

public class TesTugas2 {

    public static void main(String args[]) {

        boolean status;

        Truk truk = new Truk(900);

        System.out.println("Muatan maksimal = " + truk.getMuatanMaks());

        status = truk.tambahMuatan(500.0);

        System.out.println("Tambah muatan : 500");

        if (status)

            System.out.println("Ok");

        else

            System.out.println("Gagal");

        status = truk.tambahMuatan(300.0);

        System.out.println("Tambah muatan : 300");

        if (status)

            System.out.println("Ok");

        else

            System.out.println("Gagal");

        status = truk.tambahMuatan(150.0);

        System.out.println("Tambah muatan : 150");

        if (status)

            System.out.println("Ok");

        else

            System.out.println("Gagal");

        status = truk.tambahMuatan(50.0);

        System.out.println("Tambah muatan : 50");

        if (status)

            System.out.println("Ok");

        else

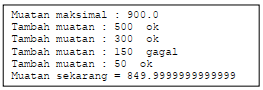
            System.out.println("Gagal");

        System.out.println("Muatan sekarang = " + truk.getMuatan());

    }

}

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak  
seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali  
program anda dan lakukan hal yang sama seperti diatas.



Berikut implementasi kode program untuk UML diatas.

* *Truk.java*

1. TUGAS
2. KESIMPULAN