MODUL PRAKTIKUM

TI03209 – PEMROGRAMAN BERORIENTASI OBJEK

Disusun oleh

Eko Mailansa, S.Kom., M.T.

D3 – TEKNIK INFORMATIKA

POLITEKNIK HASNUR

BARTIO KUALA

TAHUN 2023

MODUL III

Encapsulation

IV.1. Tujuan Praktikum

Praktikum yang dilakukan pada modul 3 bertujuan untuk:

1. Mengerti dan memahami konsep dari *Encapsulation* dalam pemrograman Java
2. Mengerti dan memahami keuntungan mengimplementasikan pola *Encapsulation* dalam pemrograman Java

IV.2. Indikator Pencapaian

Setelah melakukan praktikum maka, di harapkan praktikan mampu:

1. Membuat program sederhana yang mengimplementasikan implementasi *Encapsulation*.

IV.3. Materi

Encapsulation adalah suatu proses membungkus kode dan data menjadi satu kesatuan, misalnya kapsul yang dicampur dengan beberapa obat.

Di Java kelas **Java Bean** adalah contoh kelas yang dienkapsulasi sepenuhnya.

Metode **get** mengembalikan nilai variabel dan metode **set** menetapkan nilainya.

Enkapsulasi didefinisikan sebagai pembungkusan data dalam satu unit. Ini adalah mekanisme yang mengikat kode dan data yang dimanipulasi. Cara lain untuk memikirkan enkapsulasi adalah enkapsulasi merupakan perisai pelindung yang mencegah data diakses oleh kode di luar perisai ini.

- ✓ *Technical in enkapsulation*, variabel atau data suatu kelas disembunyikan dari kelas lain dan hanya dapat diakses melalui fungsi anggota kelasnya sendiri yang mendeklarasikannya.
- ✓ *As in enkapsulation*, data dalam suatu kelas disembunyikan dari kelas lain menggunakan konsep penyembunyian data yang dicapai dengan membuat anggota atau metode suatu kelas menjadi *private* dan kelas tersebut diekspos ke pengguna akhir tanpa memberikan rincian apa pun. Di balik implementasinya dapat

menggunakan konsep abstraksi, sehingga dikenal juga dengan **kombinasi penyembunyian data dan abstraksi**.

- ✓ Enkapsulasi dapat dicapai dengan mendeklarasikan semua variabel di kelas sebagai *private* dan menulis metode *public* di kelas untuk mengatur dan men*getter* nilai variabel.
- ✓ Enkapsulasi lebih didefinisikan dengan metode ***setter*** dan ***getter***.

Keuntungan Enkapsulasi di Java:

1. ***Data hiding*:** ini adalah cara membatasi akses anggota data dengan cara menyembunyikan detail implementasi. Enkapsulasi juga menyediakan cara untuk menyembunyikan data. Pengguna tidak akan tahu tentang implementasi bagian dalam kelas. Pengguna tidak akan dapat melihat bagaimana kelas menyimpan nilai dalam variabel. Pengguna hanya akan mengetahui bahwa kita meneruskan nilai ke metode *setter* dan variabel diinisialisasi dengan nilai tersebut.
2. ***Increased Flexibility*:** dapat membuat variabel kelas hanya baca atau tulis saja tergantung pada kebutuhan. Jika ingin menjadikan variabel hanya baca maka harus menghilangkan metode *setter* seperti ***setName(), setAge(), dll*** dari program atau jika ingin menjadikan variabel hanya tulis maka harus menghilangkan metode *getter* seperti ***getName(), getAge(), dll*** dari program.
3. ***Reusebility*:** Enkapsulasi juga meningkatkan kegunaan kembali dan mudah diubah dengan persyaratan yang baru.
4. ***Testing Code is Easy*:** Kode yang dienkapsulasi mudah diuji untuk pengujian unit.
5. ***Freedom to Programmer in Implementing the detail of the system*:** Ini adalah salah satu keuntungan utama enkapsulasi yang memberikan kebebasan kepada pemrogram dalam mengimplementasikan rincian suatu sistem. Satu-satunya kendala bagi programmer adalah mempertahankan antarmuka abstrak yang dapat dilihat oleh orang luar.

   Misalnya, pemrogram menulis kode pada menu edit di GUI editor teks mungkin pada awalnya mengimplementasikan operasi *cut* dan *paste* dengan menyalin gambar layar sebenarnya ke dalam dan ke luar buffer eksternal. Nantinya, pemrogram mungkin tidak merasa puas dengan penerapan ini, karena penerapan ini tidak memungkinkan penyimpanan pilihan yang ringkas dan tidak membedakan teks dan

objek secara grafik. Jika pemrogram telah merancang antarmuka *cut* dan *paste* dengan mempertimbangkan enkapsulasi, mengalihkan implementasi dasar ke implementasi yang menyimpan teks sebagai objek teks dan grafik dalam format yang sesuai maka, tidak akan menimbulkan masalah pada fungsi yang memerlukan antarmuka dengan GUI. Dengan demikian enkapsulasi menghasilkan kemampuan beradaptasi, karena memungkinkan rincian implementasi dari bagian-bagian program dapat berubah tanpa berdampak buruk pada bagian lainnya.

Catatan: IDE standar menyediakan fasilitas untuk menghasilkan *getter* dan *setter*. Jadi, mudah **dan cepat untuk membuat kelas enkapsulasi** di Java.

Kekurangan Enkapsulasi di Java:

1. Dapat menyebabkan peningkatan kompleksitas, terutama jika tidak digunakan dengan benar.
2. Dapat mempersulit pemahaman cara kerja sistem.
3. Dapat membatasi fleksibilitas implementasi.

Manfaat Enkapsulasi

- ✓ Atribut pada suatu kelas dapat dibuat hanya baca atau tulis saja.
- ✓ Sebuah kelas dapat memiliki kendali penuh atas apa yang disimpan di atributnya.

Mengapa Mengimplementasikan Enkapsulasi?

- ✓ Kontrol yang lebih baik atas atribut dan metode kelas
- ✓ Atribut kelas dapat dibuat **read-only** (jika hanya menggunakan metode **get**), atau **write-only** (jika hanya menggunakan metode **set**)
- ✓ Fleksibel: programmer dapat mengubah satu bagian kode tanpa mempengaruhi bagian lainnya
- ✓ Peningkatan keamanan data

IV.4. Alat dan Bahan

Peralatan dan bahan-bahan yang digunakan antara lain:

- ✓ Satu set Komputer
- ✓ Aplikasi IDE Netbeans 8.2
- ✓ Modul 4 – Mata kuliah Pemrograman Berorientasi Objek

IV.5. Praktikum

1. *Encapsulation 1*

a. Class Mahasiswa

```
package Enkapsulasi;

/**
 *
 * @author DELL
 */
public class Mahasiswa {

    private int nim;
    private String nama;
    private String email;

    //Getter Method
    public int getNim() {
        return nim;
    }

    public String getNama() {
        return nama;
    }

    public String getEmail() {
        return email;
    }

    //Setter Method
    public void setNim(int newNim) {
        nim = newNim;
    }

    public void setNama(String newNama) {
        nama = newNama;
    }

    public void setEmail(String newEmail) {
        email = newEmail;
    }
}
```

b. Class Mahasiswa_Driver

```
package Enkapsulasi;

/**
 *
 * @author DELL
 */
public class Mahasiswa_Driver {

    public static void main(String args[]) {
        Mahasiswa myBio = new Mahasiswa();
        myBio.setNim(20228888);
        myBio.setNama("Nama_Anda");
        myBio.setEmail("1234@gmail.com");

        System.out.println("BIODATA " + " \n======== "
                + " \nNIM    : " + myBio.getNim()
                + " \nNama   : " + myBio.getNama()
                + " \nEmail  : " + myBio.getEmail());
    }
}
```

2. *Encapsulation 2*

a. Class BangunRuang

```
package Enkapsulasi;

/**
 *
 * @author DELL
 */
public class BangunRuang {

    int alas;
    int tinggi;

    // constructor to initialize values
    public BangunRuang(int alas, int tinggi) {
        this.alas = alas;
        this.tinggi = tinggi;
    }

    // method to calculate LuasAlas &
    public void getLuasAlas() {
        int LA = alas * tinggi;
        System.out.println("LuasAlas : " + LA);
    }
}
```

b. Class BangunRuangMain

```
package Enkapsulasi;

/**
 *
 * @author DELL
 */
public class BangunRuangMain {

    public static void main(String[] args) {

        BangunRuang br = new BangunRuang(2, 16);
        br.getLuasAlas();
    }
}
```

IV.6. Referensi

III.6.1. Utama

- ✓ Pecinovský, Rudolf, CSc. OOP – Learn Object Oriented Thinking and Programming. ISBN 978-80-904661-8-0 (paperback) & ISBN 978-80-904661-9-7 (PDF). Published in the Czech Republic by Tomáš Bruckner, Řepín – Živonín, Academic Series. 2013.
- ✓ Wiener, Richard (University of Colorado, Colorado Springs) & J. Pinson, Lewis (University of Colorado, Colorado Springs). Fundamentals of OOP and Data Structures in Java. ISBN 0 521 66220 6 hardback and eISBN 0-511-00168 -1 virtual (netLibrary Edition). PUBLISHED BY CAMBRIDGE UNIVERSITY PRESS. 2000.

III.6.2. Pendukung

- ✓ Object Oriented Programming in JAVA. Di akses pada 16 Agustus 2023. https:// www. tutorialspoint.com/object-oriented-programming-in-java/index.asp
- ✓ Object Oriented Programming (OOPs) Concept In Java - GreekforGreeks. Di akses pada 16 Agustus 2023. https:// www.google.com/amp/s/www.greekforgreeks.com/ object-oriented-programming-oops-concept-in-java/
- ✓ Object Oriented Thinking and Programming. Di akses pada 16 Agustus 2023. https://www.w3schools.in/java/java_oop.asp

IV.7. Tugas Tambahan

1. Dari program praktikum no.1, ubah atau tambahkan kode program agar nilai dari variabel **nim, nama, email** dapat di masukkan oleh *user*/pengguna ketika program di ***run***. Gunakan ***Scanner*** atau ***Buffered Reader.***
2. Dari program praktikum no.1, tambahkan kelas dengan nama dosen dan berikan atribut **nik** dengan tipe integer, **nama** dengan tipe String, **keilmuan** dengan tipe String yang nilai dari masing-masing variable harus di masukkan oleh *user*/pengguna.
3. Dari program praktikum no.2, buatlah method pada *class* **BangunRuang** untuk menghitung **volume** (1/2 * Luas Alas * Tinggi) yang nantinya akan di panggil, untuk nilai dari alas dan tinggi variable harus di masukkan oleh *user*/pengguna.

IV.8. Laporan

- ✓ Laporan "Praktikum" di tulis sesuai dengan contoh format yang telah di berikan.
- ✓ Laporan "Tugas Tambahan" di tulis sesuai dengan contoh format yang telah di berikan dan di lampirkan di setiap laporan "Praktikum" sesuai dengan Modul Praktikum.
- ✓ Isi laporan praktikum ➔ Praktikum dan Tugas Tambahan.
- ✓ *Rename* file laporan dengan format nama yang benar (nama_modul + nim + kelas) dan berikan ekstensi atau tipe file .pdf.
  (contoh: **Modul1_20202020_3B.pdf**)