

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK
PERTEMUAN KE-2
ENKAPSULASI DAN RELASI ANTAR OBJEK

TUJUAN PRAKTIKUM

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. menerapkan konsep enkapsulasi data pada program
2. memahami relasi antar objek dan menerapkannya dalam program.

TOOLS

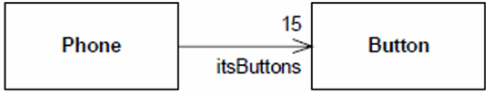
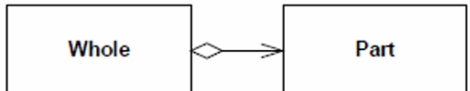
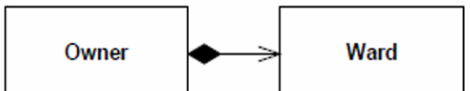
Tools yang diperlukan untuk melakukan praktikum ini adalah Java Development Kit (JDK) untuk mengkompilasi dan menjaalankan program Java serta code editor atau IDE untuk menulis program Java.

LANDASAN TEORI

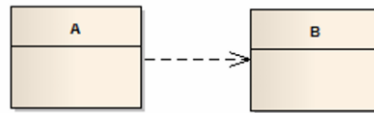
Enkapsulasi bertujuan untuk membungkus data (atribut) dan method yang memanipulasi data tersebut dalam satu blok kelas. Enkapsulasi bertujuan untuk mendukung *information hiding*, sehingga data hanya bisa diakses oleh method-method di dalam kelas tersebut. Enkapsulasi umumnya diimplementasikan dengan memberikan **access modifier private** pada data/atribut untuk memastikan data tersebut hanya bisa diakses dalam satu blok kelas, serta memberikan **access modifier public** pada konstruktor dan method lainnya yang diijinkan untuk diakses dari luar kelas. Karena atribut di-set private, maka kelas dilengkapi dengan method selektor dan mutator untuk menyediakan akses ataupun manipulasi ke nilai atribut kelas.

Objek dapat berelasi dengan objek lainnya melalui beberapa jenis relasi, yaitu pewarisan, asosiasi, agregasi, dan komposisi. Pada praktikum ini relasi yang dibahas hanyalah jenis asosiasi, agregasi, dan komposisi. Hubungan pewarisan akan dibahas pada praktikum keempat. Ketiga relasi tersebut diimplementasikan sebagai member dari class lainnya seperti yang ditunjukkan pada Tabel 1.

Tabel 1. Jenis Relasi antar Kelas dan Implementasinya

Jenis Relasi	Notasi UML	Implementasi Program Java
Asosiasi		<pre>public class Phone{ private Button itsButton[15]; }</pre>
Agregasi		<pre>public class Whole{ private Part itsPart; }</pre>
Komposisi		<pre>public class Owner{ private Ward itsWard; }</pre>

Terdapat satu jenis relasi lagi, yaitu dependensi. Dependensi adalah hubungan yang menunjukkan bahwa sebuah objek bergantung terhadap objek yang lain. Sebagai contoh kelas A menggunakan kelas B bukan sebagai member-nya, melainkan sebagai bagian proses pada program.



Gambar 1. Contoh Dependency antar Kelas

LANGKAH PRAKTIKUM

Bagian 1 - Enkapsulasi

1. Gunakan kembali file class Titik dan Garis beserta main class-nya, yaitu MTitik dan MGaris yang telah dibuat pada praktikum pertama. Simpan pada folder atau package yang baru untuk praktikum kedua.
2. Tuliskan nama file, deskripsi, pembuat, dan tanggal pada bagian awal file Anda sebagai komentar.
3. Tambahkan access modifier private pada semua atribut di dalam class Titik, lalu tambahkan access modifier public pada semua method yang ada di dalam class Titik.

```

13  public class Titik {
14  □  /*******ATRIBUT******/
15      private double absis;
16      private double ordinat;
17      private static int counterTitik = 0;
18
19  □  /*******METHOD******/
20      //konstruktor untuk membuat titik (0,0)
21
22      //konstruktor untuk membuat dengan nilai absis dan ordinat tertentu
23      public Titik(double absis, double ordinat){
24  □  this.absis = absis;
25      this.ordinat = ordinat;
26      counterTitik++;
27  }
28

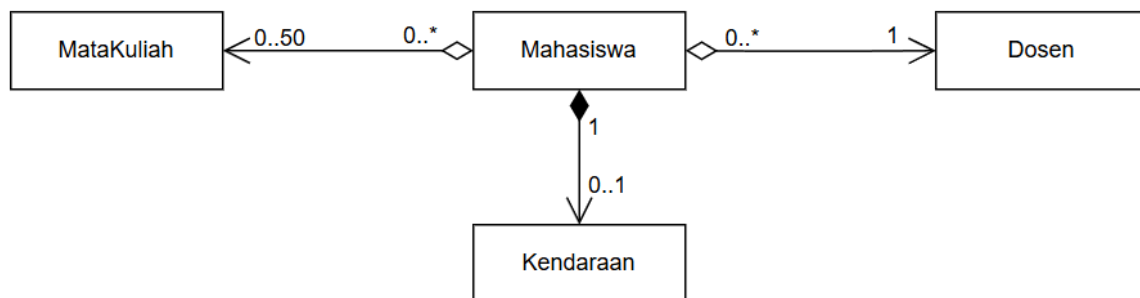
```

4. Pastikan class Titik memiliki selektor untuk setiap atributnya (absis, ordinat, dan counterTitik) serta memiliki mutator untuk atribut absis dan ordinat. Mutator untuk atribut counterTitik tidak diperlukan karena nilai atribut tersebut hanya diubah melalui konstruktor Titik pada saat ada objek yang diciptakan.
5. Pada class MTitik, buatlah sebuah objek dari class Titik serta panggilah semua atribut dan method dari objek tersebut, lalu jalankan dan amati hasilnya. Apakah perbedaan pemanggilan atribut dan method dari objek Titik pada praktikum pertama (sebelum ditambahkan access modifier) dengan praktikum ini (setelah ditambahkan access modifier)?
6. Lakukan eksperimen terhadap access modifier, misalnya ubah access modifier konstruktor dan beberapa method di class Titik menjadi private, lalu jalankan kembali class MTitik, sedemikian hingga praktikan dapat memahami perbedaan penggunaan antara access modifier public dan private.
7. Lakukan hal yang sama terhadap class Garis. Tambahkan access modifier private pada semua atributnya dan access modifier public pada semua methodnya.

8. Pada class MGaris, buatlah sebuah objek dari class Garis serta panggilah semua atribut dan method dari objek tersebut, lalu jalankan dan amati hasilnya. Apakah pemanggilan method dapat memberikan hasil sesuai yang diharapkan?

Jika tidak, periksalah kembali body method pada class Garis. Class Garis memiliki atribut yang bertipe Titik. Karena semua atribut Titik telah di-set menjadi private, maka atribut Titik tidak dapat dipanggil secara langsung di dalam class Garis. Pengaksesan nilai atribut kelas harus dilakukan melalui selektornya, sedangkan manipulasi nilai atribut kelas harus dilakukan melalui mutatornya.

Bagian 2 – Relasi antar Objek



Gambar 2. Relasi antar Kelas Mahasiswa, Dosen, MataKuliah, dan Kendaraan

Arti relasi antar kelas pada Gambar 2 tersebut adalah sebagai berikut:

1. Satu mahasiswa dapat mengambil mata kuliah paling sedikit 0 dan paling banyak 50 mata kuliah, sedangkan satu mata kuliah dapat diambil oleh 0 atau berapapun jumlah mahasiswa.
2. Satu mahasiswa harus mempunyai 1 dosen wali, sedangkan 1 dosen dapat memiliki 0 atau banyak mahasiswa perwalian.
3. Satu mahasiswa dapat memiliki 0 atau 1 kendaraan, sedangkan satu kendaraan secara eksklusif hanya dapat dimiliki oleh satu mahasiswa saja.

Lakukan Langkah-langkah sebagai berikut:

1. Buatlah class Mahasiswa, Dosen, MataKuliah, dan Kendaraan dengan ketentuan sebagai berikut:

Class	Atribut	Method
Dosen	nip, nama, dan prodi masing-masing bertipe string.	<ul style="list-style-type: none">• Konstruktor tanpa parameter.• Konstruktor dengan parameter nip, nama, dan prodi.• Selektor dan mutator untuk masing-masing atribut.
MataKuliah	idMatKul dan nama bertipe string, serta sks bertipe integer.	<ul style="list-style-type: none">• Konstruktor tanpa parameter.• Konstruktor dengan parameter idMatKul, nama, dan sks.• Selektor dan mutator untuk masing-masing atribut.

Class	Atribut	Method
Kendaraan	noPlat dan jenis bertipe string. Jenis berupa motor atau mobil.	<ul style="list-style-type: none"> Konstruktor tanpa parameter. Konstruktor dengan parameter noPlat dan jenis. Selektor dan mutator untuk masing-masing atribut.
Mahasiswa	<ul style="list-style-type: none"> nim, nama, dan prodi masing-masing bertipe string. listMatKul berupa array statik berukuran 50 bertipe MataKuliah. dosenWali bertipe Dosen. kendaraan bertipe Kendaraan. 	<ul style="list-style-type: none"> Konstruktor tanpa parameter Konstruktor dengan parameter nim, nama, prodi. Selektor dan mutator untuk masing-masing atribut. Method addMatKul() untuk menambahkan sebuah mata kuliah ke atribut listMatKul. Method getJumlahSKS() untuk mendapatkan jumlah SKS mata kuliah yang diambil mahasiswa. Method getJumlahMatKul() untuk mendapatkan jumlah mata kuliah yang diambil mahasiswa. Method printMhs() untuk menampilkan nim, nama, dan prodi mahasiswa. Method printDetailMhs() untuk menampilkan nim, nama, prodi, daftar mata kuliah yang diambil, data dosen wali, serta data kendaraan yang dimiliki mahasiswa.

Contoh Sebagian implementasi class Mahasiswa

```

11 public class Mahasiswa {
12
13     /*****ATRIBUT*****/
14     private String nim;
15     private String nama;
16     private String prodi;
17     ArrayList<MataKuliah> listMatKul;
18     private Dosen dosenWali;
19     private Kendaraan kendaraan;
20
21     /*****METHOD*****/
22
23     //konstruktor untuk membuat mahasiswa tanpa parameter
24     public Mahasiswa() {
25         this.listMatKul = new ArrayList<>(); // Inisialisasi ArrayList kosong
26     }

```

```

36 public void addMatKul (MataKuliah newMatKul) {
37     listMatKul.add(newMatKul);
38 }
39
40 public void printDetailMhs () {
41     System.out.println("Nim: " + nim);
42     System.out.println("Nama: " + nama);
43     System.out.println("Prodi: " + prodi);
44     System.out.println("Prodi: " + prodi);
45     int i;
46     for(i=0;i<listMatKul.size();i++){
47         System.out.println(listMatKul.get(i).getNama());
48     }
49 }
50 } //end class Mahasiswa

```

2. Buatlah main class MMahasiswa. Buatlah beberapa objek dari class Mahasiswa, MataKuliah, Dosen, dan Kendaraan. Relasikan setiap objek mahasiswa dengan objek dari class lainnya sesuai aturan ketentuan relasi antar class pada Gambar 2. Lalu dari objek mahasiswa panggil method untuk getJumlahSKS(), getJumlahMatKul(), dan printDetailMhs().

Contoh main method pada class MMahasiswa

```

13 public static void main(String[] args) {
14     MataKuliah PBO = new MataKuliah("PBO", "Pemrograman Berorientasi Objek", 3);
15     MataKuliah MBD = new MataKuliah("MBD", "Manajemen Basis Data", 3);
16     Mahasiswa M1 = new Mahasiswa("234", "Citra", "Informatika");
17     Dosen D1 = new Dosen("123", "Andi", "informatika");
18     Kendaraan K1 = new Kendaraan("H1234AB", "motor");
19     M1.setDosenWali(D1);
20     M1.setKendaraan(K1);
21     M1.addMatKul(PBO);
22     M1.addMatKul(MBD);
23     M1.printDetailMhs();
24     System.out.println("Jumlah Mata Kuliah = " + M1.getJumlahMatKul());
25     System.out.println("Jumlah SKS Mata Kuliah = " + M1.getJumlahSKS());
26 }

```

PELAPORAN

Selama sesi praktikum, laporkan hasil praktikum pada link <http://tiny.cc/pbo25>.

Lengkapi semua file program yang harus dikerjakan dalam modul ini dan kumpulkan hasil akhirnya di kulon maksimal H+3 setelah pelaksanaan praktikum.

*****Selamat Mengerjakan dan Berlatih *****