MODUL PERKULIAHAN

PEMROGRAMAN IV (Object I)

Class, Object, Abstraction, Encapsulation, Inheritance, Polymorphism.

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Teknik	Informatika	02		Ardiles Sinaga, S.T., M.T. Kurnia Jaya Eliazar, S.T., M.T.

Abstract	Kompetensi
Modul Pertemuan 02 Berisi Mengenai Object, Class, Abstraction, Encapsulation, Inheritance, Polymorphism.	Mahasiswa memiliki kemampuan untuk menjelaskan tentang Object, Class, Abstraction, Encapsulation, Inheritance, Polymorphisme.

Pertemuan 02 – Class, Object, Abstraction, Encapsulation, Inheritance, Polymorphism.

Class / Kelas dan Object (Objek)

Konsep Class Dan Objek

Class adalah ciri dari Object Oriented Programming. Class merupakan kumpulan dari objek yang sejenis. Sedangkan objek merupakan benda, baik secara fisik atau konseptual. Ciri dari objek adalah memiliki atribut/property/data (data member) dan method/behavior/function (member function).

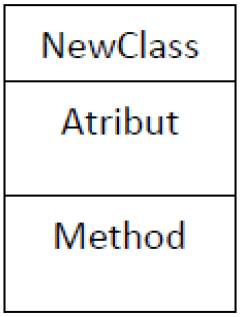
Atribut adalah variabel-variabel yang menyatakan karakteristik/ciri suatu objek (what they have). Methode adalah fungsi-fungsi yang bertugas memanipulasi nilai pada data member (what they do). Fungsi yang paling sering ada pada sebuah objek adalah fungsi untuk mengubah dan menginformasikan nilai dari data member objek. Methode juga digunakan untuk mengkomunikasikan data dalam class dengan lingkungan luar class. Pengaksesan data objek secara langsung tidak diperbolehkan.

Perbedaan antara class dengan objek adalah Class merupakan desain dan objek merupakan perwujudan suatu Class. Class bersifat abstrak dan objek bersifat kongkrit.

Class Diagram

Salah satu tools yang biasa digunakan untuk memodelkan/ menggambarkan/ merancang aplikasi berparadigma Object Oriented adalah UML atau Unified Modelling Language. Pada UML terdapat beberapa diagram yang mempunyai peran tersendiri. Salah satu diagram yang terdapat didalam UML dikenal dengan CLASS DIAGRAM. Diagram ini secara khusus menggambarkan sebuah class dan hubungan dengan class yang lain dalam sebuah sistem. CLASS DIAGRAM dan diagram-diagram yang lainnya diperkenalkan pada mata kuliah Analisis dan Perancangan Sistem Berorientasi Objek, sehingga dalam modul ini, tidak secara detail menggambarkan dan bagaimana merancang sebuah class diagram, hanya sebagai diagram konseptual untuk membantu membuat class yang sesungguhnya didalam program.

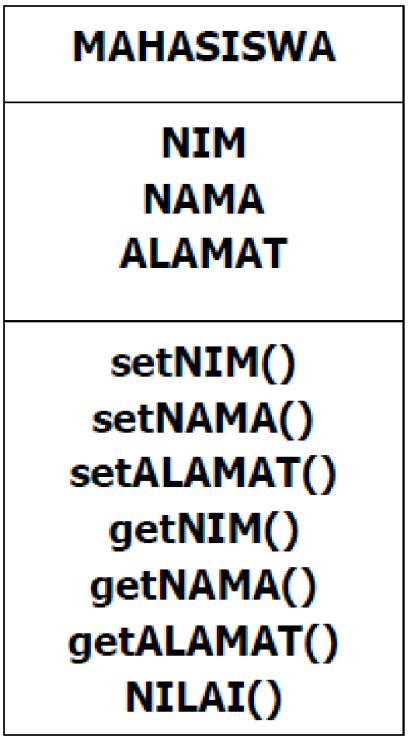
Bentuk umum dari class diagram digambarkan sebagai berikut:



Class diagram mempunyai tiga bagian. Bagian atas atau pertama digunakan sebagai nama class/kelas. Aturan membuat nama class sesuai disesuaikan dengan aturan bahasa pemrograman yang digunakan. Biasanya nama dari sebuah class sesuai dengan kumpulan object yang ditampung oleh class tersebut.
Bagian tengah atau kedua digunakan untuk menampung atribut dari class. Pada basis data atribut disebut juga sebagai FIELD. Pada bahasa java atribut disebut juga dengan INSTANCE VARIABLE.

Pada bagian yang bawah atau ketiga digunakan untuk menampung method dari class. Biasanya method berfungsi untuk memanipulasi data dari data member/atribut, tetapi tidak menutup kemungkinan sebuah method tidak berfungsi untuk memanipulasi data dari data member, tergantung dengan sifat method itu sendiri. Untuk method yang berfungsi untuk menampulasi atribut, diberi nama dengan awalan set dan get disertai dengan INSTANCE VARIABLE nya.

Contoh :
Desain sebuah class yang berfungsi sebagai kumpulan data dari mahasiswa. Class tersebut memiliki atribut diantaranya NIM, NAMA dan ALAMAT. Class tersebut juga terdapat method untuk menentukan dan mengambil data NIM, NAMA dan ALAMAT. Selain itu, juga terdapat method untuk melihat nilai yang telah diperoleh mahasiswa.
Dari uraian tersebut, class diagram untuk mahasiswa dapat didesain sebagai berikut:



Object

Objek (Object) merupakan segala sesuatu yang ada di dunia ini, yaitu manusia, hewan, tumbuhan, rumah, kendaraan, dan lain sebagainya. Contoh-contoh objek yang telah disebutkan diatas merupakan contoh objek nyata pada kehidupan kita.

Pada pemrograman berorientasi objek, kita akan belajar bagaimana membawa konsep objek dalam kehidupan nyata menjadi objek dalam dunia pemrograman. Setiap objek dalam dunia nyata pasti memiliki 2 elemen penyusunnya, yaitu keadaan (state) dan perilaku/sifat (behaviour). Sebagai contoh, sepeda memiliki keadaan yaitu warna, merk, jumlah roda, ukuran roda. Dan perilaku/sifat sepeda adalah berjalan, berhenti, belok, menambah kecepatan, mengerem.

Pada saat objek diterjemahkan ke dalam konsep PBO, maka elemen penyusunnya juga terdiri atas 2 bagian, yaitu :

Atribut, merupakan ciri-ciri yang melekat pada suatu objek (state).

Method, merupakan fungsi-fungsi yang digunakan untuk memanipulasi nilai-nilai pada atribut atau untuk melakukan hal-hal yang dapat dilakukan suatu objek (behaviour).

Objek dalam konsep PBO memiliki keadaan dan perilaku yang sama seperti halnya objek di dunia nyata, karena objek dalam konsep PBO merupakan representasi objek dari dunia nyata. Objek dalam PBO merepresentasikan keadaan melalui variabel-variabel (Atribut), sedangkan perilakunya direpresentasikan dengan method (yang merupakan suatu fungsi yang berhubungan dengan perilaku objek tersebut maupun berhubungan dengan atribut dari objek tersebut). Objek yang memiliki kesamaan atribut dan method dapat dikelompokkan menjadi sebuah Class. Dan objek-objek yang dibuat dari suatu class itulah yang disebut dengan Instant of class.

Untuk menginstansi (membuat) objek dari class, gunakan operator new.

Sintaks membuat objek dari suatu class :

```
namaClass namaObjek = new namaClass()
```

Abstraction/Abstraksi

Abstraksi adalah suatu cara melihat suatu objek dalam bentuk yang sederhana. Sebagai contoh jika kita melihat sepeda motor. Kita tidak perlu melihat susunan komponen mesin dan dukungan elektriknya yang cukup kompleks dan rumit, namun kita bisa melihat sepeda motor itu sebagai sebuah entitas / satuan tunggal (*single entity*) yang merupakan sebuah objek yang mempunyai sifat dan karakteristik tersendiri.

Dengan pemikiran yang sederhana ini maka ketika kita mengendarai sepeda motor tersebut kita tidak perlu tahu betapa rumit komponen dan rangkaian yang menyusun sepeda motor. Karena untuk mengendarai sepeda motor yang perlu diketahui adalah bagaimana sepeda motor itu bisa dikendalikan. Sehingga dengan konsep abstraksi ini kita bisa melihat suatu sistem yang kompleks yang terdiri dari subsistem-subsistem yang rumit dan banyak bisa dipandang menjadi sebuah paket sistem yang sederhana.

Pemahaman objek disekitar kita inilah yang akan mendasari pemahaman tentang pemrograman berorientasi objek. Yang paling penting adalah bagaimana mentransformasikan apa yang anda ketahui tentang suatu objek menjadi suatu program.

Encapsulation/Enkapsulasi

Enkapsulasi mempunyai beberapa nama. Ada yang menyebutnya pembungkusan, ada yang menyebutnya pengkapsulan dan ada pula yang menyebutnya *access modifier*. Mempunyai berbagai nama tetapi fungsinya adalah sama. Enkapsulasi adalah cara menyembunyikan implementasi detil sebuah class sehingga tidak bisa diakses sembarangan oleh class lainnya. Ada dua hal yang mendasar dari enkapsulasi ini adalah *INFORMATION HIDING* dan *INTERFACE TO ACCESS*.

Information Hiding adalah proses menyembunyikan data sebuah *class* sehingga tidak bisa diakses oleh *class* lain. Biasanya, data yang disembunyikan adalah atribut dari *class*, sehingga *class* yang lain tidak dapat mengetahui atribut apa saja yang dimiliki oleh sebuah *class*.

Interface to access adalah cara yang dilakukan sehingga dapat mengakses data yang telah disembunyikan. Maksudnya, yang dapat diakses adalah isi atributnya tanpa mengetahui apa saja atributnya, termasuk nama atributnya. Media yang digunakan pada cara ini biasanya adalah melalui sebuah method.

Ada 4 akses (*modifier*) yang tersedia untuk mengatur hak akses sebuah *class*. 4 akses tersebut dapat dilihat pada tabel berikut:

NO	MODIFIER	PADA CLASS DAN INTERFACE	PADA METHOD & ATRIBUT
1	Default (tak ada modifier)	Dapat diakses oleh yang sepaket.	Diwarisi oleh sub class di paket yang sama, dapat diakses oleh method-method yang sepaket
2	Public	Dapat diakses dimanapun	Diwarisi oleh sub classnya, dapat diakses dimanapun
3	Protected	Tidak bisa diterapkan	Diwarisi oleh sub classnya, dapat diakses

			oleh method-method yang sepaket.
4	Private	Tidak bisa diterapkan	Tidak dapat diakses dimanapun kecuali oleh method-method yang ada dalam kelas itu sendiri.

Apabila ditelusur dari kemampuan aksesnya, perbedaan 4 akses (modifier) tersebut dapat dilihat pada tabel berikut

AKSESABILITAS	PUBLIC	PRIVATE	PROTECTED	DEFAULT
Dari kelas yang sama	Ya	Ya	Ya	Ya
Dari sembarang kelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari sembarang kelas di luar paket	Ya	Tidak	Tidak	Tidak
Dari sub kelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari sub kelas di luar paket	Ya	Tidak	Ya	Tidak

Khusus untuk 3 akses yaitu public, private dan protected sifatnya dapat diuraikan sebagai berikut :

- Private berkarakteristik agar variabel atau method pada sebuah object/class tidak dapat diakses oleh object/class yang lain.
- Protected berkarakteristik agar variabel atau method pada sebuah object/class dapat diakses oleh object/class turunannya, tetapi tidak dapat diakses oleh object/class yang lain.
- Public berkarakteristik agar variabel atau method pada sebuah object/class dapat diakses oleh object/class yang lain

Untuk akses (modifier) Public biasanya digunakan pada method, karena method berperan sebagai Interface to Access. Method sebagai jembatan untuk mengambil data / isi data dari sebuah atribut.

Untuk akses (modifier) Private biasanya digunakan pada atribut, karena berfungsi untuk menyembunyikan data atau Information Hiding.

Untuk akses (modifier) Protected biasanya digunakan pada atribut, sehingga atribut sebuah class dapat diakses oleh beberapa class lain. Class lain yang dapat mengakses atribut adalah class yang menjadi turunan dari class tersebut.

Inheritance/Pewarisan

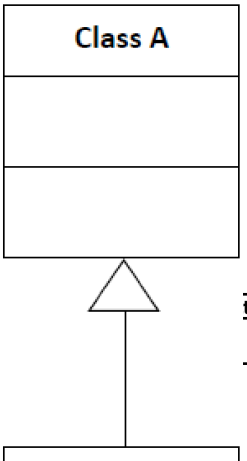
Nama lain dari Inheritance adalah Turunan atau Pewarisan. Inheritance adalah proses pewarisan data atau method dari suatu class yang sudah ada ke class yang baru. Pembuatan class baru dengan mengembangkan class yang sudah pernah dibuat sebelumnya.

Beberapa istilah dikenal pada proses inheritance diantaranya :

- Class yang menurunkan data atau method ke class lain disebut dengan super class, parent class, base class atau kelas induk.
- Class yang merupakan turunan dari kelas induk disebut dengan sub class, child class, derived class atau class turunan.

Class turunan adalah class yang menggunakan data atau method dari class induk. Konsepnya, secara otomatis class turunan memiliki sifat (variabel/atribut) dan kelakuan (behavior/method) yang sama yang dimiliki oleh kelas induknya. Class turunan dapat menambahkan fitur atau behavior dengan mendefinisikan suatu method didalam class turunan tersebut. Salah satu keuntungan dari konsep inheritance ini adalah cukup mendefinisikan satu kali saja atribut atau method yang sama pada class induknya dan dapat digunakan di seluruh class turunannya.

Pada Diagram UML (Unified Modelling Language), konsep Inheritance digambarkan pada class diagram sebagai berikut :



Contoh class diagram disamping menunjukkan bahwa Class A adalah class induk dari Class B, sebaliknya Class B adalah

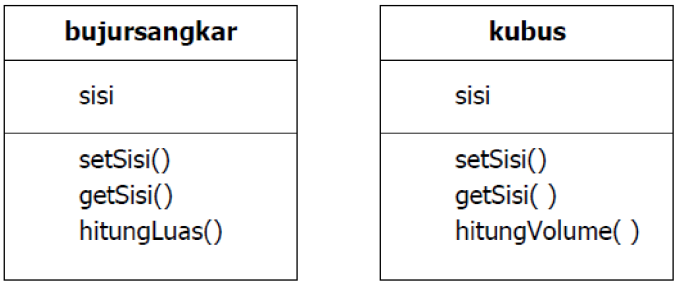
class turunan dari Class A. Panah terbuka menunjukkan bahwa kedua class tersebut menggunakan konsep Inheritance.

Manfaat dari konsep Inheritance dicontohkan sebagai berikut :

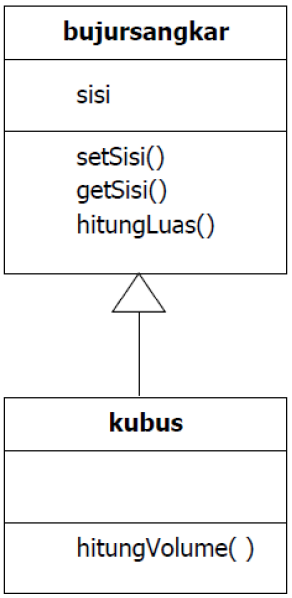
a. Buatlah sebuah class untuk menangani objek bujursangkar. Class ini diberi dengan nama bujursangkar. Atribut yang dimiliki oleh class ini adalah sisi. Untuk memanipulasi atribut tersebut, buat dua buah method dengan nama setSisi() dan getSisi(). Selain kedua method tersebut, buat juga method untuk menghitung luas sebuah objek bujur sangkar. Method tersebut dinamai dengan hitungLuas().

b. Buatlah sebuah class untuk menangani objek kubus. Class ini diberi dengan nama kubus. Atribut yang dimiliki oleh class ini adalah sisi. Untuk memanipulasi atribut tersebut, buat method dua buah method dengan nama setSisi() dan getSisi(). Selain kedua method tersebut, buat juga method untuk menghitung volume sebuah objek kubus. Method tersebut dinamai dengan hitungVolume().

Dari kedua soal tersebut, apabila dibuat dengan class diagram dapat digambarkan sebagai berikut :



Apabila diperhatikan, maka kedua class tersebut memiliki kesamaan atribut dan dua method yang sama. Sehingga kedua class ini dapat diterapkan konsep Inheritance yang digambarkan sebagai berikut :



Dengan konsep Inheritance ini, pada class kubus tidak perlu lagi dibuatkan atribut dan method yang sama dengan class bujursangkar.

class kubus dapat menggunakan atribut dan method yang sama pada class bujursangkar.

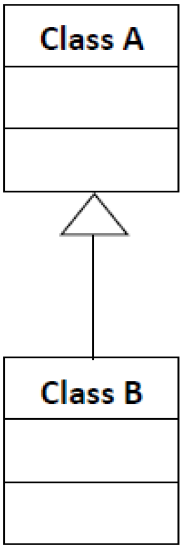
Pada pemrograman, bentuk ini dapat membantu programmer untuk meminimalkan jumlah baris perintah pada saat pembuatan program.

Macam-macam Pewarisan (Type of Inheritance)

Konsep Turunan memiliki beberapa bentuk. Bentuk dari turunan tersebut berdasarkan hubungan antara super class dan sub class. Macam-macam turunan tersebut adalah sebagai berikut :

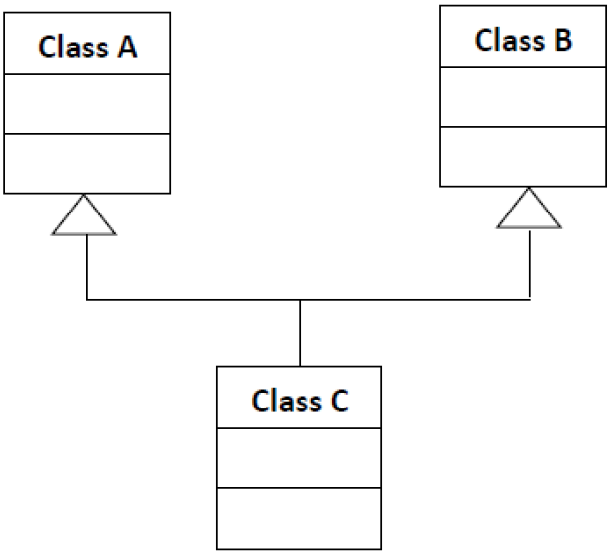
a. Single Inheritance

   Sebuah turunan disebut dengan single inheritance apabila sebuah superclass hanya memiliki satu buah subclass. Ilustrasi single inheritance digambarkan sebagai berikut



b. Multiple Inheritance

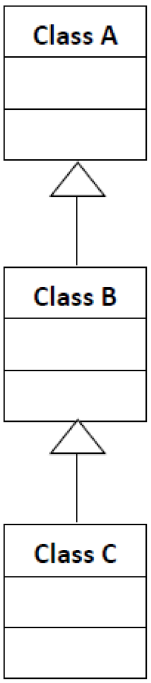
Konsep dari Multiple Inheritance adalah apabila ada sebuah sub class memiliki lebih dari satu super class. Adapun Multiple Inheritance di ilustrasikan sebagai berikut :



Berdasarkan berbagai literatur, tidak banyak bahasa pemrograman yang mendukung konsep Multiple Inheritance ini, termasuk diantaranya adalah Bahasa Pemrograman Java.

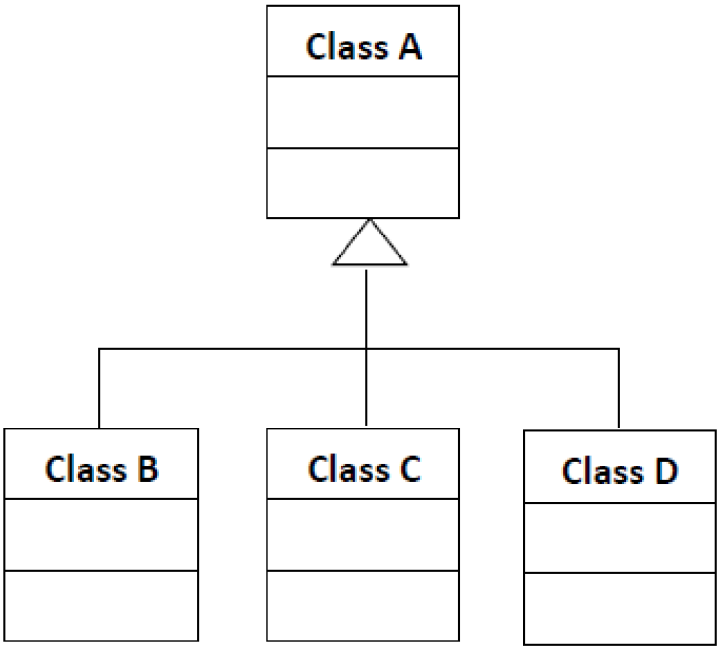
c. Multilevel Inheritance

Multilevel Inheritance terjadi apabila sebuah class menjadi sub class dari sebuah super class, tetapi kelas tersebut juga menjadi super class dari class yang lain. Ilustrasinya dapat dilihat pada gambar berikut :



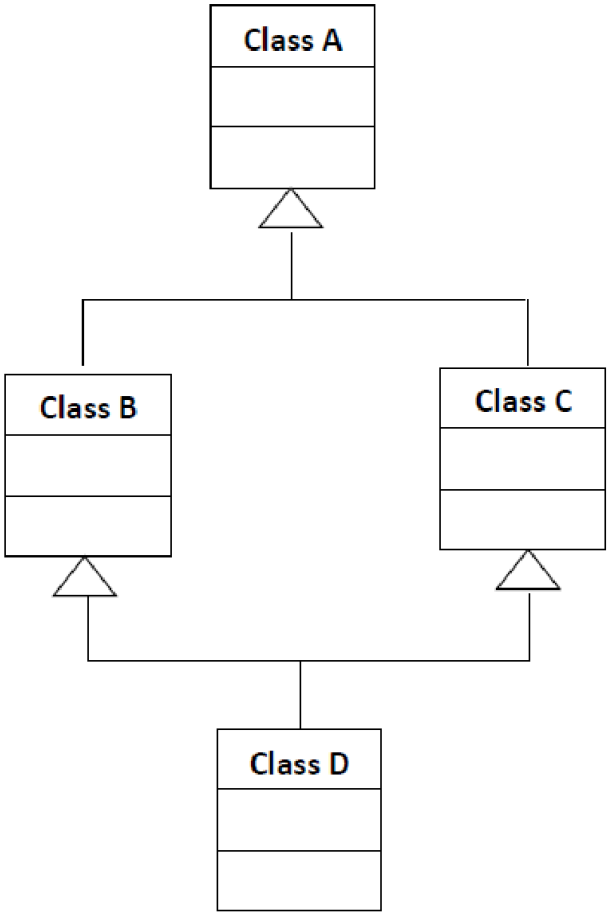
d. Hierarchical Inheritance

Bentuk turunan ini adalah apabila satu buah class yang menjadi super class memiliki lebiih dari satu sub class. Ilustrasinya dapat dilihat pada contoh berikut :



e. Hybrid Inheritance

Hybird Inheritance merupakan turunan yang merupakan kombinasi dari Single dan Multiple Inheritance. Untuk memahami konsep turunan ini, berikut ilustrasi dari Hybrid Inheritance.



Polymorphism

Polymorphism atau polimorfisme dimaknai sebagai banyak bentuk. Dalam pemrograman berorientasi objek, konsep ini memungkinkan penggunaan interface yang sama untuk keperluan atau hasil berbeda, ini yang dimaksud dengan banyak bentuk. Interface sama tetapi mempunyai hasil yang berbeda.

Ada beberapa keuntungan konsep polymorphism diterapkan didalam program diantaranya :

- Dapat menghindari duplikasi objek, dengan cara menciptakan class baru dari class yang sudah ada, sehingga tidak perlu menuliskan code dari nol ataupun mengulangnya, namun tetap bisa menambahkan atribut dan atau method unik dari class itu sendiri. Dalam

konsep yang lebih umum sering kali polymorphism disebut dalam isitlah satu interface dengan banyak aksi.

- ❖ Dapat menggunakan class-class yang dibuat sebagai super class dan membuat class-class baru berdasarkan super class tersebut dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki oleh super class.
- ❖ Dapat membuat super class yang hanya mendefinisikan behaviour namun tidak memberikan implementasi dari method-method yang ada. Hal ini berguna apabila ingin membuat semacam template class, class semacam ini disebut dengan abstrak class. Pembahasan ini akan dibahas secara khusus pada modul berikutnya.

Pada pemrograman berorientasi objek, interface yang dapat diterapkan konsep polymorphisme adalah method. Atribut sendiri tidak bisa diterapkan konsep polymorphisme. Sehingga, polymorphisme adalah konsep yang memungkinkan penggunaan method yang sama dengan hasil yang berbeda. Ada dua penyebutan nama method yang diterapkan konsep polymorphisme sesuai dengan bentuk penggunaannya. Dua method tersebut disebut dengan:

METHOD OVERLOADING dan **METHOD OVERRIDING**

Lebih jauh mengenai perbedaan kedua method tersebut dan contoh penggunaannya pada java akan diuraikan dibawah ini.

Memahami Overloading

Perhatikan penggalan baris perintah pada class berikut ini:

```
mahasiswa()
{
    setNama("Okki");
}
mahasiswa(String Nm)
{
    setNama(Nm);
}
```

Penggalan baris perintah tersebut menunjukkan terdapat dua buah konstruktor yaitu konstruktor mahasiswa() pada sebuah class. Apabila diperhatikan, maka kedua konstruktor tersebut memiliki nama yang sama, yaitu sama-sama dengan nama mahasiswa. Hal inilah yang membedakan antara teknik pemrograman terstruktur dengan teknik pemrograman berorientasi objek.

Pada pemrograman terstruktur, tidak diizinkan apabila dalam satu program terdapat lebih dari satu method dengan nama yang sama. Berbeda dengan pemrograman berorientasi objek, diperbolehkan mempunyai lebih dari satu method dengan yang sama, dengan syarat perlakuan berbeda. Cara ini disebut dengan **OVERLOADING**.

Selain pada konstruktor, konsep overloading dapat diterapkan pada beberapa method pada sebuah class dengan syarat perlakuan berbeda. Perlakuan berbeda dari setiap method ditunjukkan dengan jumlah parameter digunakan pada setiap methodnya. Selain dengan jumlah parameternya, tipe data yang tidak sama yang digunakan pada parameter juga bisa dilakukan konsep overloading walaupun jumlah parameternya sama.

Memahami Overriding

Method Overriding biasanya diterapkan pada konsep Inheritance atau turunan. Sesuai dengan fungsinya, yang dimaksud dengan overriding adalah penggunaan method sub class sama dengan method super class. Pengertian yang lain adalah, pada sub class, nama sebuah method maupun parameter method nya sama dengan nama dan parameter sebuah method pada super class. Tetapi hasil akhirnya/perlakuannya berbeda.

Inilah yang membedakan antara method Overloading dengan method Overriding. Pada method overloading, sebuah class mempunyai lebih dari satu method dengan nama yang sama tetapi parameternya yang berbeda, yang menyebabkan hasil akhirnya/perlakuannya akan berbeda.

Daftar Pustaka

[1] Deitel P.J., Deitel H.M., “Java: How to Program”, Prentice Hall, 2004

[2] Holmes B. J., Joyce D.T., “Object-Oriented Programming With Java, Second Edition”, JONES AND BARTLETT PUBLISHERS, 2001

[3] Keogh J., “JAVA DEMYSTIFIED”, McGraw-Hill/Osborne, 2004

[4] Wu C.T., “AN INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING WITH JAVA™, FIFTH EDITION”, McGraw-Hill, 2010

[5] Wu C.T., “A COMPREHENSIVE INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING WITH JAVA”, McGraw-Hill, 2008

[6] Poo D., Kiong D., Ashok S., “Object-Oriented Programming and Java Second edition”, Springer, 2008

[7] Sintes T., “Sams Teach Yourself Object Oriented Programming in 21 Days”, Sam Publishing, 2002