**Pengertian Pemrograman Berorientasi Objek**

**Pemrograman berorientasi objek** (*Object Oriented Programming* atau disingkat OOP) adalah paradigma pemrograman yang berorientasikan kepada objek yang merupakan suatu metode dalam pembuatan program, dengan tujuan untuk menyelesaikan kompleksnya berbagai masalah program yang terus meningkat. Objek adalah *entitas* yang memiliki atribut, karakter (*bahavour*) dan kadang kala disertai kondisi (*state*) (Douglas, 1992).

[](https://3.bp.blogspot.com/-NTiHPB6o20o/V_-6X-9qerI/AAAAAAAAD-4/tW_JgYUZch4TKdD7BH4FhoF-zrsaxUSqwCLcB/s1600/Pemrograman%2Bberorientasi%2Bobjek.jpg)

Pemrograman berorientasi objek ditemukan pada Tahun 1960, dimana berawal dari suatu pembuatan program yang terstruktur (*structured programming*). Metode ini dikembangkan dari bahsa C dan Pascal. Dengan program yang terstruktur inilah untuk pertama kalinya kita mampu menulis program yang begitu sulit dengan lebih mudah.

Ide dasar pada OOP adalah mengkombinasikan data dan fungsi untuk mengakses data menjadi sebuah kesatuan unit yang dikenal dengan nama objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untuk merancang aplikasi dan program komputer. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.  
  
Pemrograman berorientasi objek dalam melakukan pemecahan suatu masalah tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh sebuah departemen yang memiliki seorang manager, sekretaris, petugas administrasi data dan lainnya. Jika manager ingin memperoleh data dari bagian administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.  
  
Pemrograman berorientasi objek bekerja dengan baik ketika dibarengi dengan *Objek-Oriented Analysis And Design Process* (OOAD). Jika membuat program berorientasi objek tanpa OOAD, seperti membangun rumah tanpa terlebih dahulu penganalisis apa saja yang dibutuhkan oleh rumah itu, tanpa perencanaan, tanpa *blue-print*, tanpa menganalisis ruangan apa saja yang diperlukan, beberapa besar rumah yang akan dibangun dan sebagainya.

**BACA JUGA**

* [**Knowledge Management (Pengertian, Fungsi, Komponen, Jenis, Level dan Siklus)**](https://www.kajianpustaka.com/2020/09/Knowledge-Management.html)
* [**Enterprise Resource Planning (ERP)**](https://www.kajianpustaka.com/2020/07/enterprise-resource-planning-erp.html)
* [**Data Warehouse (Karakteristik, Komponen, Arsitektur dan Fungsi)**](https://www.kajianpustaka.com/2019/12/data-warehouse-karakteristik-komponen-arsitektur-dan-fungsi.html)
* [**Business Intelligence (Pengertian, Manfaat, Jenis, Arsitektur dan Penggunaan)**](https://www.kajianpustaka.com/2019/12/business-intelligence.html)
* [**Nomophobia (Pengertian, Aspek, Ciri, Penyebab dan Cara Menanggulangi)**](https://www.kajianpustaka.com/2021/02/nomophobia-pengertian-aspek-ciri.html)

**Konsep Dasar Pemrograman Berorientasi Objek**

|  |
| --- |
| [Konsep Dasar Pemrograman Berorientasi Objek](https://1.bp.blogspot.com/-HhNTJPKePRE/V_-6hzGqTLI/AAAAAAAAD-8/kgwGwO-yc8UKN17Ed66RR02AqEbJjelsgCLcB/s1600/Konsep%2BOOP.jpg) |
| Konsep Dasar Pemrograman Berorientasi Objek |

**a. Kelas (class)**

Kelas (class) merupakan penggambaran satu set objek yang memiliki atribut yang sama. Kelas mirip dengan tipe data ada pemrograman non objek, akan tetapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Kelas baru dapat dibentuk lebih spesifik dari kelas ada umumnya.kelas merupakan jantung dalam pemrograman berorientasi objek.

**b. Objek (Object)**

Objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan teknik yang efektif dalam menemukan cara yang tepat dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek.  
Objek adalah entitas yang memiliki atribut, karakter dan kadang kala disertai kondisi. Objek mempresentasikan sesuai kenyataan seperti siswa, mempresentasikan dalam bentuk konsep seperti merek dagang, juga bisa menyatakan visualilasi seperti bentuk huruf (font).

**c. Abstaksi (Abstraction)**

Kemampuan sebuah program untuk melewati aspek informasi yang diolah adalah kemampuan untuk fokus pada inti permasalahan. Setiap objek dalam sistem melayani berbagai model dari pelaku abstrak yang dapat melakukan kerja, laporan dan perubahan serta berkomunikasi dengan objek lain dalam sistem, tanpa harus menampakkan kelebihan diterapkan.

**d. Enkapsulasi (pembungkus)**

Pembungkusan merupakan penggabungan potongan-potongan informasi dan perilaku-perilaku spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi sesuatu yang disebut objek (Nugroho,2005).  
Enkapsulasi adalah proses memastikan pengguna sebuah objek tidak dapat menggantikan keadaan dari sebuah objek dengan cara yang tidak sesuai prosedur. Artinya, hanya metode yang terdapat dalam objek tersebut yang diberi izin untuk mengakses keadaan yang diinginkan. Setiap objek mengakses interface yang menyabutkan bagaimana objek lainnya dapat berintegrasi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

**e. Polimorfisme**

Polimorfise merupakan suatu fungsionalitas yang diimplikasikan dengan berbagai cara yang berbeda. Pada program berorientasi objek, pembuat program dapat memiliki  berbagai implementasi untuk sebagian fungsi tertentu.

**f. Inheritas (Pewarisan)**

Konsep inheritas mempunyai fungsi mengatur polimorfise dan enkapsulasi dengan mengizinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada. Objek-objek ini dapat membagi dan memperluas perilaku mereka tanpa mengimplementasikan perilaku tersebut.

**Daftar Pustaka**

* Douglas, Downing dan Michael Covington. 1992. ***Kamus Istilah Komputer***. Jakarta: Erlangga.
* Nugroho, Adi. 2005. ***Analisis dan Perancangan Sistem Informasi Dengan Metodologi Berorientasi Objek***. Bandung: Informatika.

# Pemrograman Berorientasi Objek

  15 Votes

**SILABUS MATA KULIAH**

|  |  |
| --- | --- |
| **Program Studi** | : Teknik Informatika |
| **Nama Mata Kuliah** | : Pemrograman Berorientasi Objek |
| **Jumlah SKS** | : 3 (Tiga) |
| **Semester** | : 5 |
| **Mata Kuliah Pra Syarat** | : Dasar Pemrograman Komputer |

**Deskripsi Mata Kuliah**           :

**Pemrograman berorientasi objek** (Inggris: **object-oriented programming** disingkat **OOP**) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

**Standar Kompetensi**   :

Mahasiswa dapat Mengerti Ciri dan Konsep Dasar dari Pemrograman Berbasis Objek serta Mampu Menerapkannya Untuk Membuat Berbagai Software Aplikasi Sederhana.

|  |  |  |
| --- | --- | --- |
| Kompetensi Dasar | Pengalaman Pembelajaran | Materi Ajar |
| Memahami sistem komputer dan paradigma pemrograman berbasis objek | 1. Menjelaskan sistem komputer sebagai pengolah data2. Menjelaskan komponen sistem komputer  3. Menjelaskan jaringan komputer  4. Menjelaskan proses pembuatan software  5. Menjelaskan konsep pemrograman berorientasi objek  6. Memberi contoh soal untuk dianalisa | 1. Pengenalan Sistem Komputer 2. Komponen Sistem Komputer 3. Jaringan Komputer 4. Program Development 5. Pemrograman Berorientasi Objek |
| Mampu mengenal objek di dalam pemrograman berorientasi objek dan  memahami berbagai tipe data dasar di dalam java | 1. Menjelaskan konsep objek di dalam java2. Menjelaskan tipe data dasar yang dimiliki java  3. Menjelaskan cara membuat (menurunkan) sebuah objek  4. Mendemokan contoh program untuk dianalisa | 1. Pengenalan Objek2. Pengenalan Tipe Data Dasar pada Java  3. Membuat Objek |
| Mengenal dan memahami  berbagai instruksi dasar di dalam java | 1. Menjelaskan Operator Aritmatika, Logika dan Penugasan2. Menjelaskan aturan dan instruksi Pengambilan Keputusan  3. Menjelaskan Perintah Perulangan (loop)  4. Mendemokan contoh program untuk dianalisa | 1. Operator Aritmatika, Logika dan Penugasan2. Pengambilan Keputusan  3. Perintah Perulangan (loop) |
| Mampu menunjukkan pentingnya anatomi class dan method untuk pembuatan/penulisan suatu class | 1. Menjelaskan struktur atau anatomi dari class di dalam java2. Menjelaskan struktur atau anatomi dari method di dalam java  3. Menjelaskan beberapa macam relasi antar objek  4. Mendemokan contoh program untuk dianalisa | 1. Anatomi Class2. Anatomi Method  3. Relasi Antar Objek |
| Memahami sifat dan aturan lanjut dalam pembuatan class | 1. Menjelaskan static modifier pada class2. Menjelaskan class yang bersarang (nested classes)  3. Menjelaskan interface  4. Menjelaskan Event dan Listener  5. Mendemokan contoh program untuk dianalisa | 1. Static Modifier2. Class bersarang (nested)  3. Interface  4. Events dan Listener |
| Menunjukkan pentingnya membahas array dari tipe data primitif dan objek, serta class vektor | 1. Menjelaskan array2. Menjelaskan array yang berisi objek  3. Menjelaskan class vektor | 1. Array2. Array Suatu Objek  3. Class Vektor |
| Menunjukkan pentingnya membahas cara-cara pembuatan subclass | 1. Menjelaskan definisi dan cara pembuatan subclass2. Menjelaskan sifat dan aturan overriding method  3. Menjelaskan tingkatan class | 1. Pembuatan Subclass2. Overriding Methods  3. Tingkatan Class (Class Hierarcy) |
| Menunjukkan pentingnya membahas polymorphism class | 1. Menjelaskan polymorphism class2. Menjelaskan interface hierarchies  3. Memberi contoh soal untuk dianalisa | 1. Polymorphism Class2. Interface Hierarchies |
| Menunjukkan pentingnya membahas Exception dan Input/Output Streams | 1. Menjelaskan beberapa statement dan tipe exception di dalam java2. Menjelaskan Character dan Byte Streams  3. Menjelaskan Library untuk Standard I/O  4. Memberi contoh soal untuk dianalisa | 1. Exception2. Input/Output Streams  3. Standard I/O |
| Menunjukkan pentingnya membahas GUI di dalam OOP sebagai interface visual | 1. Menjelaskan pengantar tentang GUI di dalam Javal2. Menjelaskan komponen SWING dan AWT sebagai library dasar pembuatan GUI  3. Menjelaskan fungsi dan cara pembuatan container  4. Menjelaskan fungsi dan cara pembuatan beberapa komponen di dalam GUI semisal tombol, label, dll  5. Menjelaskan manajemen pemempatan komponen  6. Memberi contoh soal untuk dianalisa | 1. Pengenalan GUI pada Java2. Container  3. Component  4. Layout Manager |
| Menunjukkan pentingnya membahas proses pengembangan suatu software | 1. Menjelaskan berbagai model pengembangan perangkat lunak2. Menjelaskan perancangan awal berupa prototype dari perangkat lunak yang akan dikembangkan  3. Menjelaskan tahapan dalam proses pengujian perangkat lunak  4. Memberi contoh soal untuk dianalisa | 1. Model Penngembangan Software2. Pembuatan Prototype  3. Pengetesan |
| Menunjukkan pentingnya membahas penerapan OOP untuk mengimplementasikan berbagai struktur data | 1. Menjelaskan konsep Link List sebagai struktur dasar dalam implementasi struktur data2. Menjelaskan implementasi Queues dan Stacks dengan Java  3. Memberi contoh soal untuk dianalisa | 1. Proses implementasi struktur data dengan java2. Implementasi Queues, dan Stacks |

ADVERTISEMENT

REPORT THIS AD

===================================================

**CATATAN**

**TUGAS INDIVIDU**

– Buat Makalah Dengan Tulis Tangan Diatas Kertas Hvs

**TUGAS KELOMPOK**

– Buat makalah dengan diketik dalam kertas A4

– Membuat presentasi dalam  powerpoint

– Tugas kelompok dikumpul lewat amalat email : ekaumaga77@*gmail.com*

====================================================

**PERBEDAAN PEMROGRAMAN BERORIENTASI OBJEK DAN PEMROGRAMAN TERSTRUKTUR**

**Pengertian Pemrograman Berorientasi Objek**

Pemrograman berorientasi objek (Inggris: object-oriented programming disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

[Bahasa pemrograman](http://id.wikipedia.org/wiki/Bahasa_pemrograman) yang mendukung OOP antara lain:

1. [Visual Foxpro](http://id.wikipedia.org/w/index.php?title=Visual_Foxpro&action=edit&redlink=1)
2. [Java](http://id.wikipedia.org/wiki/Java)
3. [C++](http://id.wikipedia.org/wiki/C_plus_plus)
4. [Pascal (bahasa pemrograman)](http://id.wikipedia.org/wiki/Pascal_%28bahasa_pemrograman%29)
5. [Visual Basic.NET](http://id.wikipedia.org/wiki/Visual_Basic.NET)
6. [SIMULA](http://id.wikipedia.org/w/index.php?title=SIMULA&action=edit&redlink=1)
7. [Smalltalk](http://id.wikipedia.org/wiki/Smalltalk)
8. [Ruby](http://id.wikipedia.org/wiki/Ruby)
9. [Python](http://id.wikipedia.org/wiki/Python)
10. [PHP](http://id.wikipedia.org/wiki/PHP)
11. [C#](http://id.wikipedia.org/wiki/C)
12. [Delphi](http://id.wikipedia.org/wiki/Delphi)
13. [Eiffel](http://id.wikipedia.org/w/index.php?title=Eiffel&action=edit&redlink=1)
14. [Perl](http://id.wikipedia.org/wiki/Perl)
15. [Adobe Flash AS 3.0](http://id.wikipedia.org/w/index.php?title=Adobe_Flash_AS_3.0&action=edit&redlink=1)

**Pengertian Pemrograman Terstruktur**

Pemrograman Terstruktur adalah suatu proses untuk mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program.  
Selain pengertian diatas Pemrograman Terstruktur adalah suatu aktifitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis , dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami.

Prinsip dari pemrograman terstruktur adalah Jika suatu proses telah sampai pada suatu titik / langkah tertentu , maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya / kembali lagi ke baris sebelumnya, kecuali pada langkah – langkah untuk proses berulang (Loop).

Bahasa pemrograman yang mendukung pemrograman terstruktur:

1. Cobol Turbo Prolog
2. C
3. Pascal
4. Delphi
5. Borland Delphi

Setelah mengetahu pengertiannya?

dilihat dari pengertian di atas, pemrograman terstruktur memilki beberapa sifat – sifat seperti :  
a. Memuat teknik pemecahan masalah yang logis dan sistematis  
b. Memuat algoritma yang efisien, efektif dan sederhana  
c. Program disusun dengan logika yang mudah dipahami  
d. Tidak menggunakan perintah GOTO  
e. Biaya pengujian program relatif rendah  
f. Memiliki dokumentasi yang baik  
g. Biaya perawatan dan dokumentasi yang dibutuhkan relatif rendah

berdasarkan penjelasan diatas, sangat jelas sekali bahwa pemrograman tersktruktur unggul dalam melakukan pemrograman sederhana karena lebih efisien dan lebih murah dalam hal perawatannya tetapi permodelan ini lebih susah untuk dipahami oleh orang – orang selain pembuat program itu sendiri (contohnya ketika dlakukan tracing program ).

Sementara itu pemrograman berorientasi objek memliki beberapa keuntungan seperti :  
1.maintenance; program lebih mudah dibaca dan dipahami, dan pemrograman berorientasi obyek mengontrol kerumitan program hanya dengan mengijinkan rincian yang dibutuhkan untuk programmer.

2.pengubahan program (berupa penambahan atau penghapusan fitur tertentu); perubahan yang dilakukan antara lain menyangkut penambahan dan penghapusan dalam suatu database program misalnya.

3.dapat digunakannya obyek-obyek sesering yang diinginkan, kita dapat menyimpan obyek-obyek yang yang dirancang dengan baik ke dalam sebuah tolkit rutin yang bermanfaat yang dapat disisipkan kedalam kode yang baru dengan sedikit perubahan atau tanpa perubahan pada kode tersebut.

jadi, sangat jelas sekal bahwa pemrograman berorientasi objek sangat cocok sekali digunakan dalam kasus pembuatan software yang rumit dan kompleks karena memberikan berbagai kemudahan kepada pemrogram seperti yang telah disebutkan diatas.

permodelan yang mana yang lebh bagus? itu tergantung dari kebutuhan dan dari sudut pandang mana anda melihatnya. Yang perlu anda ingat adalah tujuan dari pemodelan itu sendiri, yang mana agar pada akhir proyek sistem dapat diperoleh sistem informasi yang memenuhi kebutuhan pemakai, tepat waktu dan sesuai anggaran, serta mudah digunakan, dimengerti dan dipelihara

Perbedaan mendasar antara OOP dan pemrograman terstruktur adalah:

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sedangkan untuk pemrograman terstruktur,  menggunakan prosedur/tata cara yang teratur untuk mengoperasikan data struktur

Untuk tata nama, keduanya pun memiliki tatanan yang sama walaupun memiliki pengertian tersendiri:

object oriented menggunakan “method” sedangkan terstruktur menggunakan “function”. Bila di OOP sering didengar mengenai “objects” maka di terstruktur kita mengenalnya dengan ” modules”. Begitu pula  halnya dengan “message” pada OO dan “argument” pada terstruktur. “attribute” pada OO juga memiliki tatanan nama yang sepadan dengan “variabel” pada pemrograman terstruktur.

Persamaan kedua pemrogaman

adalah keduanya termasuk ke dalam pemodelan pemrograman yang digunakan dewasa ini.

Mana yang lebih baik

tidak ada yang dapat dikatakan lebih baik karena keduanya memiliki spesifikasi tersendiri dalam pemrogramannya. Hal ini juga tergantung pada bagaimana pribadi si pemrogram ingin menyusun program yang akan dibuatnya. Apakah lebih suka menggunakan yang berorientasikan pada objek maupun pemrograman yang terstruktur.

Kapan sama2 baik

Pemrograman prosedural akan dikatakan lebih baik apabila dalam segala situasi melibatkan kompleksitas moderat atau yang memerlukan signifikan kemudahan maintainability. Manfaat yang dirasakan dalam penggunaan pemrograman prosedural adalah kemampuan kembali menggunakan kode yang sama tanpa menggunakan kode yang berbeda ataupun mengkopinya kembali. Dengan menggunakan “goto”, memudahkan programmer melacak kumpulan data sehingga menghindarkan pemrograman terstruktur menjadi seperti spagethii code.

Pemrograman berorientasikan objek dikatakan lebih baik apabila Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

## [Konsep Dasar Pemrograman Berorientasi Objek](http://www.adityarizki.net/2012/06/konsep-dasar-pemrograman-berorientasi-objek/)

**Prosedural vs OOP**

Dalam dunia pemrograman, biasanya kita dihadapkan pada dua jenis metode pemrograman, yaitu **pemrograman prosedural (procedural)** dan **pemrograman berorientasi objek (object oriented)**. Apa perbedaan kedua metode tersebut? Bagaimana pemrograman berorientasi objek itu dilakukan? Apa saja konsep-konsep OOP yang harus dikuasai? Tulisan ini akan menjelaskan pertanyaan-pertanyaan tersebut.

Pemrograman prosedural merupakan suatu metode menulis program yang didasarkan pada**“serangkaian tugas yang diselesaikan dalam bentuk fungsi atau prosedur”**. Cara pandang pemrograman prosedural yaitu sebuah program adalah suatu urutan instruksi. Programmer harus mem-break down suatu problem/masalah menjadi sub problem yang lebih sederhana. Fokus utama metode prosedural ini adalah fungsi dan prosedur, dimana keduanya digunakan untuk memanipulasi data. Dalam hal ini data bersifat pasif.

Lain halnya dengan pemrograman berorientasi objek (OOP), fungsi dan data bukan menjadi dua hal yang terpisah. Fungsi dan data menjadi satu kesatuan yang disebut sebagai objek aktif. Cara pandang OOP ini yaitu sebuah program merupakan **serangkaian objek yang bekerjasama untuk menyelesaikan suatu problem**.

Dengan kata lain, metode prosedural berfokus pada cara komputer menangani tugas, sedangkan metode OOP berfokus pada tugas yang kita kembangkan untuk dieksekusi komputer. Kedua jenis metode pemrograman tersebut dapat digunakan untuk menangani masalah yang sama, asalkan bahasa pemrograman yang digunakan mendukung metode-metode tersebut. Contoh bahasa pemrograman yang mendukung OOP diantaranya : Java, C++, Pascal, Visual Basic .NET, Ruby, Python, PHP, C#, Delphi, Perl, dsb. Selain itu masih banyak bahasa lain yang termasuk ke dalam bahasa prosedural, atau bahkan bisa mendukung kedua metode pemrograman tersebut. Untuk lebih lengkapnya, dapat dilihat daftar bahasa pemrograman berdasarkan tipenya di [sini](http://en.wikipedia.org/wiki/List_of_programming_languages_by_type).

**Istilah-Istilah OOP**

**1. Objek**

Untuk mempermudah pemahaman, maka disini akan dijelaskan melalui analogi. Pada dasarnya semua benda yang adadi dunia nyata dapat dianggap sebagai objek. Misalnya rumah, mobil, sepeda, motor, gelas, komputer, meja, sepatu, dll. Setiap objek memiliki atribut sebagai status (state) dan tingkah laku sebagai behavior.

Contoh objek : Motor. Maka atribute (state) nya adalah pedal, roda, jeruji, speedometer, warna, jumlah roda. Sedangkan tingkah laku (behavior) nya adalah kecepatan menaik, kecepatan menurun, dan perpindahan gigi motor.

Analogi pemrograman berorientasi objek sama dengan penggambara pada dunia nyata seperti contoh di atas. Dalam OOP, state disimpan pada variabel dan tingkah laku disimpan pada method.

Dalam bahasa teoretis OOP, Objek berfungsi untuk membungkus data dan fungsi bersama menjadi satu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

**2. Class**

Definisi class yaitu template untuk membuat objek. Class merupakan prototipe atau blue prints yang mendefinisikan variabel-variabel dan method-method secara umum. Objek merupakan hasil instansiasi dari suatu class. Proses pembentukan objek dari suatu kelas disebut sebagai instantiation. Objek disebut juga sebagai instances.

Dalam bahasa teoretis OOP, class merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh ‘class of dog’ adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object.

Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

**3. Attributes**

Atribut adalah data yang membedakan antara objek satu dengan yang lainnya.

Contoh Objek : VolcanoRobot ( a volcanic exploration vehicle), mempunyai atribut sebagai berikut:

* Status ~> exploring, moving, returning home
* Speed ~> in miles per hour
* Temperature ~> in Fahrenheit degrees

Dalam class, atribut sering disebut sebagai variabel. Atribut dibedakan menjadi dua jenis yaitu Instance Variable dan Class Variable.

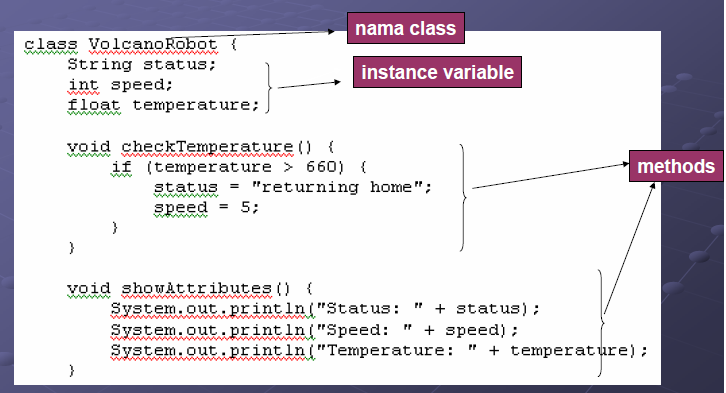
Instance variable adalah atribut untuk tiap objek dari kelas yang sama. Tiap objek mempunyai dan menyimpan nilai atributnya sendiri. Jadi, tiap objek dari class yang sama bokeh mempunyai nilai yang sama atau berbeda.

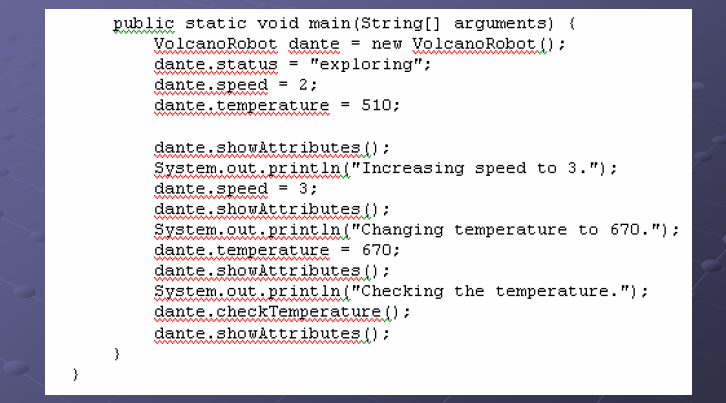
Class Variable adalah atribut untuk semua objek yang dibuat dari class yang sama. Semua objek mempunyai nilai atribut yang sama. Jadi semua objek dari class yang sama mempunyai hanya satu nilai yang value nya sama.

**4. Behavior**

Behavior/tingkah laku adalah hal-hal yang bisa dilakukan oleh objek dari suatu class. Behavior dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke objek lain untuk melakukan suatu tugas (task).

Contoh behavior untuk objek VolcanoRobot:

[](https://informartitekuswim.wordpress.com/materi-kuliah/pemrograman-berorientasi-objek/image001/)

[](https://informartitekuswim.wordpress.com/materi-kuliah/pemrograman-berorientasi-objek/image003/)

* check current temperature
* begin a survey
* report its current position

Dalam class, behavior disebut juga sebagai methods. Methods sendiri adalah serangkaian statements dalam suatu class yang menghandle suatu task tertentu. Cara objek berkomunikasi dengan objek yang lain adalah dengan menggunakan method.

Contoh class, object, attributes, dan behavior:

**5. Abstraksi**

Abstraksi adalah kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari “pelaku” abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

**Konsep Konsep OOP**

**1. Enkapsulasi (Encapsulation)**

Definisi enkapsulasi: Pembungkusan variabel dan method dalam sebuah obyek yang terlindungi serta menyediakan interface untuk mengakses variabel tersebut. Variabel dan method yang dimiliki oleh suatu objek, bisa ditentukan hak aksesnya. Dalam OOP, konsep enkapsulasi sebenarnya merupakan perluasan dari struktur dalam bahasa C.

Contoh: jam tangan. Dalam hal ini, penting sekali untuk mengetahui waktu, sedangkan cara jam mencatat waktu dengan baik antara jam bertenaga baterai atau bertenaga gerak tidaklah penting kita ketahui.

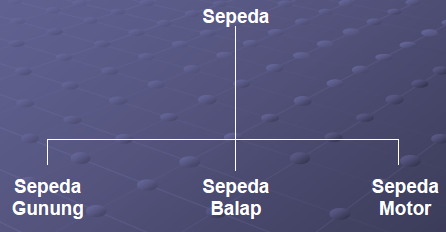
Dengan kata lain enkapsulasi berfungsi untuk memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam/dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

**2. Pewarisan (Inheritance)**

Pewarisan merupakan pewarisan atribut dan method dari sebuah class ke class lainnya. Class yang mewarisi disebut superclass dan Class yang diwarisi disebut subclass. Subclass bisa berlaku sebagai superclass bagi class lainya, disebut sebagai multilevel inheritance.

Contoh : terdapat class sepeda dan sepeda gunung. Sepeda termasuk superclass.  Sepeda gunung termasuk subclass. Hal ini dikarenakan sepeda gunung memiliki variabel dan method yang dimiliki oleh sepeda.

Prinsip dasar inheritance yaitu persamaan-persamaan yang dimiliki oleh beberapa kelas dapat digabungkan dalam sebuah class induk sehingga setiap kelas yang diturunkannya memuat hal-hal yang spesifik untuk kelas yang bersangkutan.

[](https://informartitekuswim.wordpress.com/materi-kuliah/pemrograman-berorientasi-objek/image005/)

Contoh Pewarisan

**Keuntungan Pewarisan**

* Subclass menyediakan state/behaviour yang spesifik yang membedakan dengan superclass, sehingga memungkinkan programmer untuk menggunakan ulang source code dari superclass yang telah ada.
* Programmer dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut abstract class (abstraksi), untuk mendefinisikan class dengan tingkah laku dan state secara umum.

**Single & Multiple Inheritance**

Bahasa C adalah contoh multiple inheritance. Suatu class diperbolehkan untuk mempunyai lebih dari satu superclass. Variabel dan method yang diwariskan merupakan kombinasi dari superclass-nya. Java adalah contoh single inheritance. Suatu class hanya boleh mempunyai satu superclass.

**Multilevel Inheritance**

Suatu subclass bisa menjadi superclass bagi class yang lain.

**3. Polimorfisme**

Polimorfisme adalah kemampuan suatu obyek untuk mempunyai lebih dari satu bentuk. Polimorfisme tidak bergantung kepada pemanggilan subrutin. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan “gerak cepat”, dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

**Kesimpulan :**  
Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bagian administrasi, maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah, dapat dilakukan dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

**Resources:**

* Wikipedia
* Slide Presentasi Konsep Pemrograman Berorientasi Objek (*lecturer.eepis-its.edu/~riyanto/oop1.pdf*)

**MENGENAL TIPE DATA DAN VARIABEL DALAM PEMROGRAMAN**

Salah satu komponen utama program adalah variabel. Variabel adalah objek di mana informasi dinamis disimpan. Ketika Anda menuliskan nama Anda saat hendak membuat account baru atau proses login ke website, nama yang Anda masukkan akan disimpan dalam variabel sebelum diproses lebih lanjut.

Tidak semua variabel memiliki karakteristik yang sama. Informasi yang dapat disimpan oleh variabel ditentukan berdasarkan jenis atau tipe data variabel terebut. Terdapat berbagai jenis tipe data dalam setiap bahasa pemrograman. Namun, kali ini kita akan melihat beberapa jenis tipe data yang umumnya ada pada hampir seluruh bahasa pemrograman. Tipe-tipe data tersebut antara lain sebagai berikut :

**Tipe data Karakter (character/char)**

Jenis tipe data ini hanya memungkinkan sebuah variabel menyimpan informasi sebuah single karakter, seperti huruf “P”,, “c”, “7”, atau “@”.

**Tipe data String**

Tipe data string ini memungkinkan variabel menyimpan informasi untaian karakter, seperti kata atau kalimat. Dalam contoh sebelumnya, nama yang Anda gunakan pada web form, informasi tersebut akan disimpan dalam variabel bertipe data string.

**Tipe data Integer (int)**

Jenis tipe data integer digunakan jika ingin menyimpan informasi angka (bilangan bulat) dalam variabel, tetapi bukan bilangan pecahan. Hal ini berarti nilai “7” akan menjadi nilai integer yang valid.

**Tipe data Floating Point Number (float)**

Jika Anda bermaksud menyimpan informasi angka dengan format pecahan atau angka desimal, floating point adalah jenis tipe data yang sesuai. Angka “65,00” secara teknis merupakan nilai bertipe floating point meskipun nilai tersebut juga dapat direpresentasikan sebagai nilai “65”. Jenis tipe data ini membutuhkan ukuran memory yang lebih besar dibandingkan jenis tipe data angka integer.

**Tipe data Boolean (bool)**

Jenis tipe data ini biasanya digunakan hanya untuk merepresentasikan dua kondisi, yakni nilai TRUE dan FALSE atau daatberarti bernilai 1 dan 0 atau kondisi ON dan OFF. Jenis tipe data ini merupakan jenis yang paling sederhana, teapi cukup sering digunakan dalam setiap pemrograman.

**Tipe data Array**

Jenis tipe data aray sering disebut juga sebagai tipe data larik. Tipe data ini pada dasarnya merupakan kumpulan sejumlah variabel bertipe data sama dengan ukuran tertentu, yang tersusun secara runtun. Ada beragam variasi dari tipe data array, bergantung pada bahasa pemrograman yang digunakan. Sebagai contoh, nilai “1”, “2”, “3”, “4”, dan “5” dapat Anda simpan ke dalam variabel bertipe data array dengan ukuran aray “5” (5 buah nilai, mulai dari 1 sampai dengan 5) yang memiliki kumpulan nilai bertipe integer atau secar sederhana dapat dikatakan array of integer values.

Setiap variabel yang terdapat dalam array dapat diakses dengan menggunakan komponen indeks dari tiap-tiap variabel trseut. Namun, perlu Anda ketahui bahwa variabel pertama yang ada dalam array bukan memiliki indeks 1, melainkan indeks 0. Menyimpan nilai-nilai tersebut ke dalam variabel bertipe data array akan memudahkan kit untuk melakukan manipulasi terhadap setiap data yang ada dalam array tersebut. perlu diketahui bahwa jenis tipe data string sebenarnya merupakan kumpulan variabel bertipe karakter atau bisa dikatakan array of characters.

Bottom of Form