OOP (Object oriented Programing)

OOP adalah paradigma pemrograman yang cukup dominan saat ini, karena mampu memberikan solusi kaidah pemrograman modern. Meskipun demikian, bukan berarti bahwa pemrograman prosedural sudah tidak layak lagi .  
  
OOP diciptakan karena dirasakan masih adanya keterbatasan pada bahasa pemrograman tradisional. Konsep dari OOP sendiri adalah, semua pemecahan masalah dibagi ke dalam objek. Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Proses perancangan atau desain dalam suatu pemrograman merupakan proses yang tidak terpisah dari proses yang mendahului, yaitu analisis dan proses yang mengikutinya. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti inheritance atau penurunan, encapsulation atau pembungkusan,   
dan polymorphism atau kebanyakrupaan. Konsep-konsep ini merupakan fundamental dalam orientasi objek yang perlu sekali dipahami serta digunakan dengan baik, dan menghindari penggunaannya yang tidak tepat.  
Class dan Objek  
Dalam lingkungan program berorientasi objek, pemrogram mendefinisikan class secara statik. Pada saat run-time, class akan diinstantiasi menjadi objek. Ada pun objek yang merupakan instantiasi dari suatu class selalu dapat diacu melalui current objek, apa pun nama instant-nya.  
Dapat didefinisikan bahwa class merupakan struktur data dari suatu objek, lebih jelasnya adalah sebuah bentuk dasar atau blueprint yang mendefinisikan variabel method umum pada semua objek dari beberapa macam. Objek sendiri adalah kumpulan variabel dan fungsi yang dihasilkan dari template khusus atau disebut class. objek adalah elemen pada saat run-time yang akan diciptakan, dimanipulasi, dan dihancurkan ketika eksekusi. Ada pun class merupakan definisi statik dari himpunan objek yang mungkin diciptakan sebagai instantiasi dari class. Sederhananya adalah kumpulan objek yang mempunyai atribut sama. Dengan demikian, pada saat run-time maka yang kita miliki adalah objek.  
Class biasany dibuat seperti ini  
Class nama\_classnya{  
//berisi sekumpulan fungsi,variabel dan  
//statement-statement lainnya  
}  
Agar class tersebut dpt digunakan maka perlu memiliki atribut-atribut contonya dalam php/c++  
//di php  
class NamaClass {  
var $atribut1;  
var $atribut2;  
function operasi(){}  
}  
//di c++  
class NamaClass {  
public:  
int atribut1;  
int atribut2;  
int operasi(){retun nilai\_int}  
};  
Pada bahasa pemerograman web lain ex. Java file harus disimpan dengan nama class berbeda dengan PHP kita dapat memberikan nama yang tidak mesti sama dengan nama classnya  
Inheritance  
Inheritance atau pewarisan adalah kalimat yang pasti ada di dalam pemerograman berorientasi objek disebut juga dengan istilah reusable.  
Ketika kita menggunakan kembali atau mengganti method dari class yang sudah ada, serta ketika menambahkan field instant dan method baru, maka pada saat itulah Anda bekerja dengan inheritance. Konsep ini merupakan konsep yang fundamental dalam orientasi objek dan harus digunakan dengan baik, ada beberapa macam Inheritance atau pewarisan yang ada di dalam OOP Single Inheritance dan Multple Inheritance  
- Single Inheritance  
  
- Multiple Inheritance  
  
dalam penggunaan inheritance class Induk akan mewariskan semua atribut yang ia miliki sehingga class bawahnya akan memilki atribut yang sama dengan class induk, bahkan dapat memilki kebabasan untuk memilki atribut berbeda dengan class induknya berikut ilustrasi dalam php  
catatan :  
sampai sekarang PHP belum mendukung multiple inheritance seperti C++ walapun PHP dan c/c++ begitu banyak kemiripannya, dan perubahan PHP versi 5 ditekankan pada OOP salah satunya adalah beberapa tambahan untuk kemampuan inheritance  
/\* inheritance.php \*/  
class Bapak {  
var $nama =”Bapak”;  
function Bapak($n) {  
$this->nama = $n;  
}  
function Hallo() {echo “Halo, saya $this->nama  
”;}  
}  
class Anak extends Bapak {}  
$test = new Anak(”Anak dari Bapak”);  
$test->Hallo();  
?>  
Hasil tampilan dari listing program di atas adalah “Halo, saya Anak dari Bapak” dan bukannya “Halo, saya Bapak”. Mengapa demikian? Memang di dalam class Bapak didefinisikan variabel nama dengan nilai Bapak, selanjutnya kita membuat objek dari class Anak yang merupakan turunan dari class Bapak. Lihat bahwa instantiasi sekaligus mengisikan parameter baru “Anak dari Bapak”, sehingga ketika dipanggil maka mengisikan $this->nama dengan parameter tersebut. Ada pun di dalam implementasi pemrograman, kebanyakan pemrogram merasakan beberapa manfaat dari inheritance atau pewarisan, diantaranya:  
Subclass mampu menyediakan perilaku khusus dari elemen dasar yang disediakan oleh superclass.  
Pemrogram dapat mengimplementasikan superclass untuk memanggil class abstrak yang menyatakan perilaku umum.

Pemograman adalah Suatu kumpulan urutan pemrintah ke computer untuk mengerjakan sesuatu,dimana intruksi tersebut menggunakan bahasa yang   
  
dimengerti oleh computer atau di kenal dengan bahasa pemograman.  
  
Konsep Objec Oriented sebenarnya bukanlah barang baru dalam dunia bahasa pemograman. Konsep ini sudah lama ada di mulai dari bahasa pemograman Small Talk,LISP,C++,dan beberapa bahasa pemogaran lainnya.  
  
Pemoograman Object Oriented di maksudkan adalah membuat suatu program yang terdiri dari berbagai object yang saling berinteraksi. Dalam mebangun program yang masih sederhana konsep object oriented ini akan terasa sangat menyusahkan, akan tetapi ketika kita membuat Sebuah program yang ruaang lingkupnya luas konsep object oriented akan terasa sangat membantu karena akan terasa lebih mudah untuk mengorganisir program yang telah kita buat.  
  
Dalam konsep object oriented akan kita temukan kaata object dn class,class merupakan pola atau teamplate yang menggambarkan kumpulan object yang mempunyai sifat yang sama,perilaku,atu disebut dengan himpunan object sejenis. Sementara object adalah implementasi dari class. Sebagai contoh : dalam pembangunan suatu gedung tentunya seorang arsitek akaan menggunakan gambar desain gedung,dan kemudian akan di buat gedung nyatanya, Dalam contoh ini gambar desain gedung dapat kita kategorikan sebagai class dan gedungnyata sebagai implementasi dari gambar gedung yang dapat kita kata gorikan sebagai object,Dari suatu class dapat di buat beberapa object yang sama.   
  
  
• OOP (Object oriented Programing)  
  
Pemograman Object Oriented adalah membuat suatu program yang terdiri dari berbagai oject yang saling beriteraksi (dalam beberapa buku di artikan dengan bertukar pesan antar object). Ketika membuat Sebuah program yang ruang lingkupnya luas konsep object oriented akan terasa sangat membantu karna akan terasa lebih mudah untuk menggorganisir program yang telah kita buat,Dalam konsep object oriented terdapat 2 istilah yaitu objeck dan class,class merupakan pola atau template yang menggambarkan kumpulan object yang mempunyai sifat yang sama,perilaku,atau disebut dengan himpunan object sejenis. Sementara object adalah implementasi dari class. *Tujuan utama dari pengembngan perangkat lunak atau program berorientasi oject adalah   
  
1. Mempersingkat waktu dan menurunkan biaya pengembangan   
Rekayasa perangkat lunak   
  
2. Menurunkan biaya perwatan perangkat lunak  
  
3. Pemograman beririentasi object memberikan landasan yang sangat berguna untuk pembuatan prototype system secara cepat.*  
  
Pedoman yang di sajikaan untuk menentukan Object-object dalam suatu proram:  
  
a Correctness Sufficiency  
  
  
Dalam mendesain proram sebaiknya kita memperhatikan bahwa object-object yang kita desain dapat menjamin kebenaran informasi dan juga kecukupan informasi yang di butuhkan.  
  
a Robustness   
  
  
Objetc-object yang kita desain sebaiknya dapat mebuat suatu program menjadi tangguh,dalam hal ini di harapkan program yang di buat meiliki error yang sedikit dan dapat bertahan dalam situasi yang sulit sekalipun.  
  
a Flexibility  
  
  
Object –object yang didesain sebaiknya mudah untuk di expand atau di kembangkan (bersifat flexbiliaty). Hal ini di butuh kan karena dalam perbuatan program sering kali kita harus merubah desain yang telah kita buat sebelumnya karena adanya perubahan desain yang di ingn kan oleh user ataupengguna.  
  
  
a Reasbility  
  
  
Kode-kde pogram yang telah kita buat,dapat kita gunakan kembali.Hal ini haruslah terpenuhi dalam mendesain program yang baik. Karena ketika kita menggunakan point ini maka kita akan lebih mudah untuk me-manage program yang kita buat.  
  
  
  
a Efficiency   
  
Desain yang kita buat sebaiknya efesien karena akan berkaitan dengan penggunaan resorce hardware yang kita gunakan. Baik itu dalam factor kecepatan maupun dalam factor penyimpanan data atau program.  
  
  
  
• Pemrograman berorientasi Object (OOP)  
  
Pemrograman berorientasi Object merupakan paradigma pemrograman yang menggunakan " objek "- struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka - untuk merancang aplikasi dan program komputer. Programming techniques may include features such as data abstraction , encapsulation , modularity , polymorphism , and inheritance . Pemrograman teknik dapat mencakup fitur seperti data abstraksi , enkapsulasi , modularitas , polimorfisme , dan warisan . Many modern programming languages now support OOP. Banyak modern bahasa pemrograman sekarang mendukung OOP.  
• ikhtisar   
Sebuah objek adalah bundel diskrit fungsi dan prosedur, seringkali berkaitan dengan konsep dunia nyata tertentu seperti pemegang rekening bank, pemain hoki, atau buldoser. Lain buah perangkat lunak dapat mengakses objek hanya dengan memanggil fungsi dan prosedur yang telah diizinkan untuk dipanggil oleh orang luar Sejumlah besar insinyur perangkat lunak setuju bahwa mengisolasi objek dengan cara ini membuat software mereka lebih mudah untuk mengelola dan melacak. Namun, sejumlah besar insinyur merasa kebalikannya mungkin benar: bahwa perangkat lunak menjadi lebih kompleks untuk mempertahankan dan dokumen, atau bahkan untuk insinyur dari awal Kondisi di mana OOP menang atas teknik alternatif (dan sebaliknya) sering tetap tak tertulis oleh salah satu pihak, bagaimanapun, pembahasan rasional dari topik yang sulit, dan sering menyebabkan perdebatan sengit [ rujukan? ] atas masalah.   
Pemrograman berorientasi obyek memiliki akar yang dapat ditelusuri ke tahun 1960-an. Sebagai perangkat keras dan software menjadi semakin kompleks, pengelolaan sering menjadi perhatian. Para peneliti mempelajari cara untuk menjaga kualitas software dan pemrograman berorientasi objek yang dikembangkan sebagian untuk mengatasi masalah-masalah umum dengan sangat menekankan diskrit, unit dapat digunakan kembali logika. Teknologi ini berfokus pada data daripada proses, dengan program yang terdiri dari modul mandiri ("kelas"), setiap contoh yang ("objek") berisi semua informasi yang dibutuhkan untuk memanipulasi data struktur sendiri ("anggota"). Hal ini berbeda dengan yang ada pemrograman modular yang telah dominan selama bertahun-tahun yang difokuskan pada fungsi dari sebuah modul, bukan data spesifik, tetapi juga disediakan untuk penggunaan kembali kode , dan cukup dapat digunakan kembali unit-diri dari logika pemrograman, memungkinkan kolaborasi melalui penggunaan modul terkait ( subrutin ).. Pendekatan yang lebih konvensional, yang masih tetap, cenderung untuk mempertimbangkan data dan perilaku secara terpisah.   
Istilah "benda" dan "berorientasi" dalam sesuatu seperti pengertian modern berorientasi objek pemrograman tampaknya membuat penampilan pertama mereka di MIT pada akhir 1950-an dan awal 1960-an "member functions Dalam lingkungan dari kecerdasan buatan kelompok, pada awal 1960, "objek" bisa merujuk ke item diidentifikasi ( LISP atom) dengan sifat (atribut); Alan Kay kemudian mengutip pemahaman rinci internal LISP sebagai pengaruh yang kuat pada pemikirannya pada tahun 1966. Contoh lain MIT awal Sketchpad diciptakan oleh Ivan Sutherland pada tahun 1960-1961, dalam glossary dari laporan teknis 1963 berdasarkan disertasi tentang Sketchpad, Sutherland didefinisikan pengertian dari "objek" dan "contoh" (dengan konsep kelas ditutupi oleh "master" atau "definisi"), walaupun khusus untuk interaksi grafis. [4] Juga, sebuah MIT ALGOL versi, AED-0, terkait struktur data ("plexes", di dialek) secara langsung dengan prosedur, pertanda apa yang kemudian disebut "pesan", "metode" dan "fungsi anggota".   
Objek sebagai konsep formal dalam pemrograman diperkenalkan pada tahun 1960 di Simula 67, revisi utama dari Simula I, bahasa pemrograman yang dirancang untuk simulasi kejadian diskrit , diciptakan oleh Ole-Johan Dahl dan Kristen Nygaard dari Norwegia Computing Center di Oslo . [ 7] Simula 67 dipengaruhi oleh SIMSCRIPT dan Hoare's diusulkan "kelas merekam". [5] [8] Simula memperkenalkan gagasan kelas dan contoh atau benda (dan juga subclass, metode virtual, coroutines, dan simulasi kejadian diskrit) sebagai bagian sebuah paradigma pemrograman yang eksplisit. The language also used automatic garbage collection that had been invented earlier for the functional programming language Lisp . Bahasa ini juga digunakan otomatis pengumpulan sampah yang telah diciptakan sebelumnya untuk pemrograman fungsional bahasa Lisp . Simula yang digunakan untuk pemodelan fisik, seperti model-model untuk mempelajari dan meningkatkan pergerakan kapal dan isinya melalui pelabuhan kargo . Ide-ide Simula 67 dipengaruhi bahasa banyak kemudian, termasuk Smalltalk, turunan dari LISP ( CLOS ), Object Pascal , dan C + + .   
The Smalltalk bahasa, yang dikembangkan di Xerox PARC (oleh Alan Kay dan lain-lain) pada 1970-an, memperkenalkan pemrograman berorientasi obyek istilah untuk mewakili penggunaan meresap objek dan pesan sebagai dasar untuk perhitungan. . Smalltalk pencipta dipengaruhi oleh ide-ide yang diperkenalkan dalam Simula 67, tapi Smalltalk dirancang untuk menjadi sistem yang dinamis sepenuhnya di mana kelas dapat dibuat dan dimodifikasi secara dinamis, bukan statis seperti dalam Simula 67. [9] Smalltalk dan dengan itu OOP diperkenalkan untuk khalayak yang lebih luas oleh Agustus 1981 edisi majalah Byte .   
Pada 1970-an, Smalltalk karya tersebut Kay telah mempengaruhi masyarakat Lisp untuk menggabungkan teknik-teknik berbasis objek yang diperkenalkan untuk pengembang melalui mesin Lisp . . Eksperimen dengan berbagai ekstensi Lisp (seperti Loops dan Rasa memperkenalkan multiple inheritance dan mixin ), akhirnya mengarah pada Common Lisp Object System (CLOS, bagian dari berorientasi obyek bahasa pemrograman standar pertama, ANSI Common Lisp ), yang mengintegrasikan pemrograman fungsional dan pemrograman berorientasi obyek dan memungkinkan perpanjangan melalui protokol Meta-objek . the Linn Smart Rekursiv . Contoh termasuk iAPX Intel 432 dan Linn Smart Rekursiv .   
Pemrograman berorientasi obyek yang dikembangkan sebagai metodologi pemrograman dominan di awal dan pertengahan 1990-an ketika bahasa pemrograman yang mendukung teknik menjadi tersedia secara luas. . Ini termasuk Visual FoxPro 3.0 C + + dan Delphi. dominasi semakin ditingkatkan oleh meningkatnya popularitas antarmuka pengguna grafis , yang sangat tergantung pada teknik pemrograman berorientasi-objek.. Sebuah contoh dari GUI dinamis perpustakaan erat kaitannya dan bahasa OOP dapat ditemukan di Kakao kerangka pada Mac OS X , ditulis dalam Objective-C , sebuah berorientasi, dinamis-ekstensi objek pesan ke C berdasarkan Smalltalk.. OOP toolkit juga meningkatkan popularitas pemrograman event-driven (walaupun konsep ini tidak terbatas pada OOP).. Beberapa merasa bahwa hubungan dengan GUI (nyata atau dianggap) adalah apa yang didorong OOP ke dalam arus pemrograman.   
, Niklaus Wirth dan rekan-rekannya juga telah menyelidiki topik seperti abstraksi data dan pemrograman modular (walaupun ini telah umum digunakan pada tahun 1960 atau sebelumnya). Modula-2 (1978) termasuk baik, dan berhasil desain mereka, Oberon , termasuk pendekatan khusus untuk objek orientasi, kelas, dan semacamnya Pendekatan ini tidak seperti Smalltalk, dan sangat tidak seperti C + +.   
Obyek fitur telah ditambahkan ke banyak bahasa yang ada selama waktu itu, termasuk Ada , BASIC , Fortran , Pascal , dan lain-lain. Menambahkan fitur ini untuk bahasa yang awalnya tidak dirancang untuk mereka sering menyebabkan masalah dengan kompatibilitas dan rawatan kode.   
Baru-baru ini, sejumlah bahasa telah muncul yang terutama berorientasi objek belum kompatibel dengan metodologi prosedural, seperti Python dan Ruby .. Mungkin komersial penting baru-baru ini bahasa berorientasi objek yang paling adalah Visual Basic.NET (VB.NET) dan C # , baik yang dirancang untuk Microsoft NET. platform, dan Jawa , yang dikembangkan oleh Sun Microsystems .kedua kerangka menunjukkan manfaat penggunaan OOP dengan membuat sebuah abstraksi dari implementasi dengan cara mereka sendiri.. VB.NET dan C # mendukung warisan lintas bahasa, kelas memungkinkan didefinisikan dalam satu bahasa ke bahasa subclass kelas didefinisikan dalam bahasa lain. Java berjalan dalam mesin virtual, sehingga memungkinkan untuk berjalan di semua sistem operasi yang berbeda.. VB.NET dan C # menggunakan pola Strategi untuk mencapai-bahasa warisan lintas, sedangkan Jawa memanfaatkan pola Adapter   
Sama seperti pemrograman prosedural menyebabkan perbaikan teknik seperti pemrograman terstruktur , desain perangkat lunak berorientasi objek metode modern meliputi perbaikan [ rujukan? ] seperti penggunaan pola desain , desain dengan kontrak , dan bahasa pemodelan (seperti UML ).   
  
• Konsep dasar dan fitur   
Sebuah survei oleh Deborah J. Armstrong hampir 40 tahun sastra komputasi mengidentifikasi sejumlah "quark", atau konsep mendasar, ditemukan dalam mayoritas kuat definisi dari OOP.   
Tidak semua konsep-konsep ini dapat ditemukan dalam semua bahasa pemrograman berorientasi-objek, dan pemrograman berorientasi objek yang menggunakan kelas kadang-kadang disebut pemrograman berbasis kelas . tidak biasanya menggunakan kelas. Akibatnya, secara signifikan berbeda namun terminologi analog digunakan untuk mendefinisikan konsep objek dan contoh.   
Cuire Benjamin Pierce dan beberapa peneliti lain melihat sebagai sia-sia setiap usaha untuk menyaring OOP untuk satu set minimal fitur Meskipun demikian, ia mengidentifikasi fitur fundamental yang mendukung gaya pemrograman OOP dalam bahasa berorientasi objek yang paling   
• Dynamic dispatch - ketika metode bisa dipanggil pada objek, obyek itu sendiri menentukan kode apa yang dijalankan dengan melihat metode pada waktu berjalan dalam sebuah tabel yang berhubungan dengan objek. Fitur ini membedakan objek dari suatu tipe data abstrak (atau modul), yang memiliki pelaksanaan (statis) tetap operasi untuk semua kasus. Ini adalah metodologi pengembangan program yang memberikan komponen modular sementara pada saat yang sama yang sangat efisien.   
• Encapsulation (atau multi-metode , dalam hal ini negara disimpan terpisah)   
• Subtipe polimorfisme   
• Object inheritance (or delegation ) Objek warisan (atau delegasi )   
• rekursi Buka - variabel khusus (sintaktis mungkin kata kunci), biasanya disebut this atau self , yang memungkinkan badan metode untuk memanggil metode tubuh lain objek yang sama Variabel ini akhir-terikat, yang memungkinkan sebuah metode yang didefinisikan dalam satu kelas untuk memanggil metode lain yang ditentukan kemudian, dalam beberapa subclass daripadanya.   
  
Demikian pula, dalam bukunya 2003, Konsep-konsep dalam bahasa pemrograman, John C. Mitchell mengidentifikasi empat fitur utama: dinamis pengiriman, abstraksi , subtipe polimorfisme, dan pewarisan. Michael Lee Scott dalam Bahasa Pemrograman pragmatik hanya mempertimbangkan enkapsulasi, pewarisan dan dinamis pengiriman . [16]   
  
Main article: Class (computer science) Artikel utama: Kelas (ilmu komputer)   
  
. kelas adalah template untuk suatu obyek, suatu datatype ditetapkan pengguna yang berisi variabel, properti, dan metode. kelas mendefinisikan karakteristik abstrak sesuatu (objek), termasuk karakteristiknya (atributnya, ladang atau properti ) dan hal-hal yang bisa dilakukan (perilaku, metode , operasi atau fitur. Orang mungkin mengatakan bahwa kelas adalah cetak biru atau pabrik yang menggambarkan sifat sesuatu Sebagai contoh, kelas Dog akan terdiri dari ciri-ciri yang dimiliki oleh semua anjing, seperti berkembang biak dan warna bulu (karakteristik), dan kemampuan untuk kulit dan duduk (perilaku). Kelas menyediakan modularitas dan struktur dalam sebuah program komputer berorientasi-objek.. kelas A biasanya harus dikenali oleh programmer non-akrab dengan masalah domain, yang berarti bahwa karakteristik kelas harus masuk akal dalam konteks. Selain itu, kode untuk kelas harus relatif self-contained (umumnya menggunakan enkapsulasi Secara kolektif, sifat dan metode yang didefinisikan oleh kelas disebut anggota.   
Artikel utama: Instance (ilmu komputer)   
Satu dapat memiliki sebuah instance dari kelas; contoh adalah objek yang sebenarnya dibuat pada saat run-time Dalam programmer vernakular, yang Lassie objek merupakan instance dari Dog kelas. Set nilai dari atribut objek tertentu disebut nya negara . Objek ini terdiri dari negara dan perilaku yang didefinisikan di kelas objek.   
Artikel utama: Metode (ilmu komputer)   
. Metode adalah satu set pernyataan prosedural untuk mencapai hasil yang diinginkan.. Ia melakukan berbagai jenis operasi pada tipe data yang berbeda Dalam bahasa pemrograman, metode (terkadang disebut sebagai "fungsi") adalah verba. Lassie , menjadi Dog , memiliki kemampuan untuk kulit. So bark() is one of Lassie 's methods. Jadi bark() adalah salah satu Lassie metode s '. . Dia mungkin memiliki metode-metode lain juga, misalnya sit() atau eat() atau walk() atau save(Timmy) .. Dalam program tersebut, menggunakan metode biasanya mempengaruhi hanya satu objek tertentu; semua Dog s mengonggong, namun Anda hanya perlu satu anjing tertentu untuk melakukan menggonggong.   
  
Artikel utama: Message passing   
. "Proses di mana suatu objek mengirim data ke obyek lain atau meminta objek lain untuk memanggil sebuah metode." [13] Juga dikenal dengan beberapa bahasa pemrograman interfacing. Sebagai contoh, objek yang disebut Breeder dapat memberitahu Lassie objek untuk duduk dengan melewati sebuah "duduk" pesan yang memanggil Lassie's "duduk" metode. Sintaks bervariasi antara bahasa, misalnya: [Lassie sit] di Objective-C. In Java, code-level message passing corresponds to "method calling". Di Jawa, kode-Tingkat kelulusan pesan sesuai dengan "metode panggilan". Beberapa bahasa dinamis menggunakan double-pengiriman atau multi-dispatch untuk menemukan dan lulus pesan.   
Artikel utama: Abstraksi (ilmu komputer)   
. Abstraksi mengacu pada tindakan yang merupakan fitur penting tanpa termasuk rincian latar belakang atau penjelasan. Classes use the concept of abstraction and are defined as a list of abstract attributes. Kelas menggunakan konsep abstraksi dan didefinisikan sebagai daftar atribut abstrak.   
Artikel utama: Encapsulation (pemrograman berorientasi obyek)   
Enkapsulasi menyembunyikan rincian fungsional kelas dari objek yang mengirim pesan itu.   
. Sebagai contoh, Dog kelas memiliki bark() variabel metode, data Kode untuk mendefinisikan metode persis bagaimana kulit terjadi (misalnya, dengan inhale() dan kemudian, di sebuah lapangan tertentu dan volume). Timmy, teman s ', bagaimanapun, tidak perlu tahu persis bagaimana dia menyalak Enkapsulasi dicapai dengan menetapkan kelas dapat menggunakan anggota objek Hasilnya adalah bahwa setiap objek menghadapkan untuk setiap kelas tertentu antarmuka - para anggota dapat diakses oleh kelas tersebut. Alasan untuk enkapsulasi adalah untuk mencegah klien dari sebuah antarmuka dari tergantung pada bagian-bagian dari implementasi yang mungkin berubah di masa depan, sehingga memungkinkan perubahan tersebut akan dibuat lebih mudah, yaitu, tanpa perubahan untuk klien. Sebagai contoh, sebuah antarmuka dapat memastikan bahwa anak anjing hanya dapat ditambahkan ke sebuah objek dari kelas Dog oleh kode di kelas itu Anggota sering ditetapkan sebagai public, protected atau swasta, menentukan apakah mereka yang tersedia untuk semua kelas, sub-kelas atau hanya kelas mendefinisikan. Beberapa bahasa melangkah lebih jauh: Jawa menggunakan modifier akses standar untuk membatasi akses juga untuk kelas-kelas dalam paket yang sama, C # dan VB.NET cadangan beberapa anggota kelas-kelas dalam majelis yang sama menggunakan kata kunci internal (C #) atau Teman (VB.NET). Eiffel dan C + + membolehkan satu untuk menentukan kelas dapat mengakses setiap anggota.   
Artikel utama: Warisan (pemrograman berorientasi obyek)   
. Inheritance memungkinkan programmer untuk memperlakukan anggota kelas turunan seperti anggota kelas induk mereka. Jenis hubungan ini disebut anak-orang tua atau-hubungan"Subclass" adalah versi yang lebih khusus dari kelas, yang mewarisi atribut dan perilaku dari kelas orang tua mereka, dan bisa memperkenalkan mereka sendiri.   
Sebagai contoh, kelas Dog mungkin memiliki sub-kelas yang disebut Collie , Chihuahua , dan GoldenRetriever. Dalam hal ini, akan menjadi sebuah instance dari Collie subclass. Misalkan Dog kelas mendefinisikan sebuah metode yang disebut bark() dan sifat yang disebut furColor . Masing-masing sub-kelas nya ( Collie , Chihuahua , dan GoldenRetriever ) akan mewarisi para anggota, yang berarti bahwa programmer hanya perlu menulis kode untuk mereka sekali.   
. Sebagai contoh, Collie subclass mungkin menetapkan bahwa default furColor untuk collie adalah coklat-putih.. The Chihuahua subclass mungkin menetapkan bahwa metode menghasilkan nada tinggi secara default.. The Chihuahua subclass dapat menambahkan metode yang disebut tremble() Jadi sebuah contoh chihuahua individu akan menggunakan bernada tinggi bark() dari Chihuahua subclass, yang pada gilirannya mewarisi biasa bark() dari Dog .. Objek chihuahua juga akan memiliki tremble() metode, tetapi Lassie tidak mau, karena dia adalah Collie , bukan Chihuahua Pada kenyataannya, warisan adalah "sebuah ... adalah" hubungan antara kelas, sedangkan instantiation adalah sebuah "adalah" hubungan antara suatu objek dan kelas: sebuah adalah Dog ("a ... adalah"), tetapi Lassie adalah Collie ("adalah"). Dengan demikian, objek bernama Lassie memiliki metode dari kedua kelas Collie dan Dog .   
. Beberapa warisan adalah warisan dari lebih dari satu kelas leluhur, baik dari nenek moyang menjadi seorang nenek moyang yang lain.. Sebagai contoh, kelas independen dapat mendefinisikan Dog s dan Cat s, dan Chimera objek dapat dibuat dari dua yang mewarisi semua multiple) perilaku (dari kucing dan anjing.. Ini tidak selalu didukung, karena akan sulit untuk melaksanakan.   
[   
Artikel utama: polimorfisme Subtipe   
. Polimorfisme adalah proses di mana kelas memiliki semua negara dan perilaku kelas lain.   
. Lebih tepatnya, Polimorfisme dalam pemrograman berorientasi obyek adalah kemampuan benda milik berbagai jenis data untuk menanggapi panggilan dari metode dengan nama yang sama, masing-masing menurut sebuah perilaku tipe yang tepat.. Salah satu metode, atau operator seperti +, -, atau \*, bisa abstrak diterapkan dalam berbagai situasi.. Jika Dog diperintahkan untuk) , hal ini dapat mendatangkan. Namun, jika suatu Pig diperintahkan untuk, hal ini dapat menimbulkan suatu Setiap subclass menimpa metode warisan dari orang tua kelas .   
. Decoupling memungkinkan untuk pemisahan interaksi objek dari kelas dan warisan ke lapisan yang berbeda dari abstraksi Penggunaan umum dari decoupling adalah untuk memisahkan polymorphically enkapsulasi, ] yang merupakan praktek menggunakan kode dapat digunakan kembali untuk mencegah modul kode diskrit dari berinteraksi satu sama lainNamun, dalam prakteknya sering melibatkan decoupling trade-offs yang berkaitan dengan pola-pola perubahan menguntungkan. Ilmu mengukur trade-off ini sehubungan dengan perubahan yang sebenarnya secara obyektif masih dalam masa pertumbuhan   
• Definisi Formal   
Ada beberapa upaya memformalkan konsep yang digunakan dalam pemrograman berorientasi obyek. Konsep berikut dan konstruksi telah digunakan sebagai interpretasi konsep OOP:   
• datatypes coalgebraic   
• abstract data types (which have existential types ) allow the definition of modules but these do not support dynamic dispatch tipe data abstrak (yang memiliki jenis eksistensial ) memungkinkan definisi modul tetapi ini tidak mendukung pengiriman dinamis   
• recursive types records catatan menjadi dasar untuk memahami objek jika fungsi literal dapat disimpan dalam bidang (seperti dalam bahasa pemrograman fungsional), tetapi perlu bate sebenarnya jauh lebih kompleks untuk memasukkan fitur penting dari OOP. Beberapa ekstensi dari Sistem F <: yang berhubungan dengan objek bisa berubah telah dipelajari ini memungkinkan kedua polimorfisme subtipe dan polymorphism parametrik (generik)   
Upaya untuk menemukan definisi konsensus atau teori di balik benda tidak terbukti sangat sukses (Namun, lihat Abadi & Cardelli untuk definisi formal konsep OOP banyak dan konstruksi), dan sering menyimpang secara luas. Sebagai contoh, beberapa definisi berfokus pada aktivitas mental, dan beberapa program hanya penataan .. Salah satu definisi sederhana adalah bahwa OOP adalah tindakan menggunakan struktur "peta" data atau array yang dapat berisi fungsi dan pointer ke peta lainnya dengan beberapa sintaktis dan gula scoping di atas. Warisan dapat dilakukan oleh kloning peta (kadang-kadang disebut "prototipeOBYEK: =>> Objek adalah entitas run time dalam suatu sistem berorientasi objek. Mereka mungkin mewakili orang, tempat, rekening bank, tabel data atau item apapun bahwa program tersebut telah untuk menangani.   
• bahasa OOP   
  
  
Daftar bahasa pemrograman berorientasi-objek   
. Simula (1967) secara umum diterima sebagai bahasa pertama yang memiliki fitur-fitur utama dari sebuah bahasa berorientasi objek. Organisasi ini dibentuk untuk membuat program simulasi , di mana apa yang kemudian disebut objek adalah informasi perwakilan paling penting. Smalltalk (1972-1980) ini bisa dibilang contoh kanonik, dan yang satu dengan yang banyak teori pemrograman berorientasi objek adalah dikembangkan.: Mengenai derajat orientasi objek, berikut perbedaan dapat dibuat:   
• . Bahasa disebut "murni" bahasa OO, karena segala isinya diperlakukan secara konsisten sebagai objek, dari primitif seperti karakter dan tanda baca, semua jalan sampai ke seluruh kelas, prototipe, blok, modul, dll Mereka dirancang khusus untuk memfasilitasi, bahkan menegakkan, metode OO. Examples: Smalltalk, Eiffel , Ruby , JADE . Contoh: Smalltalk, Eiffel , Ruby , JADE .   
• Bahasa pemrograman yang dirancang terutama untuk OO, tapi   
• dengan beberapa elemen prosedural. Examples: C++ , C# , Java , Python . Contoh: C + + , C # , Java , Python .   
• Bahasa yang secara historis bahasa prosedural , tetapi telah diperpanjang dengan beberapa fitur OO. Examples: VB.NET (derived from VB), Fortran 2003 , Perl , COBOL 2002, PHP , ABAP . Contoh: VB.NET (berasal dari VB), Fortran 2003 , Perl , COBOL 2002, PHP , ABAP .   
• . Bahasa dengan sebagian besar fitur objek (kelas, metode, warisan, usabilitas), tetapi dalam bentuk jelas asli. Examples: Oberon (Oberon-1 or Oberon-2). Contoh: Oberon (Oberon-1 atau Oberon-2).   
• . Bahasa dengan tipe data abstrak dukungan, tetapi tidak semua fitur-orientasi obyek, kadang-kadang disebut bahasa berbasis objek. Contoh: Modula-2 (dengan enkapsulasi yang sangat baik dan menyembunyikan informasi), liat , CLU .   
• OOP dalam bahasa dinamis   
Dalam beberapa tahun terakhir, pemrograman berorientasi objek telah menjadi sangat populer di bahasa pemrograman dinamis . Python , Ruby dan Groovy adalah bahasa dinamis dibangun di atas prinsip-prinsip OOP, sedangkan Perl dan PHP telah menambahkan fitur berorientasi objek sejak Perl 5 dan PHP 4, dan ColdFusion sejak versi 6.   
. The Document Object Model dari HTML , XHTML , dan XML dokumen di Internet telah binding ke populer JavaScript / ECMAScript bahasa.. JavaScript mungkin yang paling dikenal prototipe berbasis pemrograman bahasa, yang mempekerjakan kloning dari prototipe daripada mewarisi dari kelas. Lain bahasa scripting yang mengambil pendekatan ini Lua . Versi sebelumnya dari ActionScript (superset sebagian dari R3 ECMA-262, atau dikenal sebagai ECMAScript) juga menggunakan objek berdasarkan model-prototipe. Kemudian versi ActionScript menggabungkan kombinasi klasifikasi dan model obyek berbasis prototipe sebagian besar didasarkan pada spesifikasi-262 R4 ECMA saat ini tidak lengkap, yang berakar dalam 2 JavaScript Proposal awal.. Microsoft JScript.NET juga mencakup mash-up model obyek berdasarkan proposal yang sama, dan juga merupakan superset dari spesifikasi   
Bahasa pemrograman yang mendukung OOP antara lain:  
1. Visual Foxpro  
2. Java  
3. C++  
4. Pascal (bahasa pemrograman)  
5. Visual Basic.NET  
6. SIMULA  
7. Smalltalk  
8. Ruby  
9. Python  
10. PHP  
11. C#  
12. Delphi  
13. Eiffel  
14. Perl  
  
  
• implementasi OOP dalam Java-contoh program penghitung luas segitiga  
Dalam tutorial ini tidak akan saya jelaskan secara terperinci apa maksud kata “implementasi OOP” diatas. Tentang OOP beserta konsep dasarnya, saya anggap anda sudah paham, namun jika belum, pahami terlebih dahulu dan tinggalkan artikel ini. karena dalam belajar selalu butuh tahap.  
baiklah para pembaca yang cerdas, seperti biasa, langsung ke source code nya :  
class segitigaDgnOOP  
—————————————————–begin————————————————  
public class segitigaDgnOOP  
{  
private int alas;  
private int tinggi;  
  
public void setAlas(int alas)  
{  
this.alas = alas;  
}  
public void setTinggi(int tinggi)  
{  
this.tinggi = tinggi;  
}  
public int getAlas()  
{  
return alas;  
}  
public int getTinggi()  
{  
return tinggi;  
}  
public double hitungLuas()  
{  
double luas;  
luas=0.5\*alas\*tinggi;  
return luas;  
}  
}