**Sejarah OOp**

Konsep OOp pertama kali muncul di MIT (Massachusetts Institute of Technology) pada era 1960-an. Sekitar beberapa tahun kemudian (1962-1965), sebuah bahasa pemrograman yang mendasari konsep oop diperkenalkan dengan nama bahasa pemrograman SIMULA 1, dikembangkan oleh Kristen Nygaard dan Ole-Johan yang merupakan warga norwegia. Dan pada tahun 1967 keluarlah SIMULA 67. Bahasa SIMULA dengan OOp menginspirasi banyak pengembang, sehingga pada tahun 70-an tercipta dan ter-inspirasi oleh bahasa OOp sebelumnya, yaitu bahasa SMALTALK dan menjadi yang pertama kali disebut sebagai Object-Oriented, dan membawa fitur barunya yaitu Inheritance.

Semenjak itu konsep OOp semakin terkenal, padatahun 1980-an, sekitar empat bahasa pemrograman lebih di perkenalkan, dan empat yang terkenal yaitu ADA (US Departemen of Defense), pROLOG (the Japanese “Fifth Generation Computer project”), Eifle dan C++. Kedua bahasa ADA dan pROLOG dipercaya akan bersaing ketat sebagai bahasa pemrograman yang paling dominan.

Namun, bahasa pemrograman Object-Oriented, C++ menjadi popular pada tahun 80-an sampai 90-an, Yang merupakan penggabungan dari 2 Bahasa yaitu bahasa C dan SIMULA. Semenjak C++ terkenal banyak sekali pengembang terinspirasi oleh C++ dan pada tahun 90-an, bahasa pemrograman JAVA diperkenalkan yang mengaku terinspirasi oleh C++, dan tahun 2002 perusahaan Microsoft juga mengeluarkan bahasa turunan dari C++ yaitu C# (C-Sharp), disusul dengan VB.NET dengan fitur OOp yang merupakan penyempurnaan dari bahasa VB 0.6 yang tidak mendukung fitur OOp.

**Pengertian Object Oriented programming(OOp)**

OOp atau dengan kepanjangan, Object Oriented programming adalah metodologi atau paradigma pemrograman dalam melakukan pemrograman yang beriontasi kepada object, semua data dan fungsi akan dibungkus dalam kelas-kelas dan object-object. OOp berfungsi untuk memberikan kemudahan, fleksibilitas, kemudahan mengubah program, dan perawatan dalam pengembangan program. Dan juga banyak penda pat yang mengatakan bahwa pemrograman yang berbasis object lebh mudah diperlajari oleh pemula dibandingkan dengan bahasa pemrograman yang tidak berbasis object.

OOp berfungsi untuk mempemudah dalam pengembangan program, dengan cara mengikuti model yang telah ada di dalam kehidupan sehari-hari.

OOp mudah dimengerti oleh pemula karena konsep OOp mirip seperti model yang telah ada di dalam kehidupan sehari-hari kita (Object-Oriented World). Bisa kita simpulkan bahwa dari setiap suatu permasalahan adalah object, dan object (1) tersebut merupakan gabungan dari beberapa object (2) yang lebih kecil lagi dengan fungsi (method) yang berbeda-beda dan membentuk object (1) tersebut. dalam kehidupan sehari-hari, disaat kita melakukan aktivitas, sebenarnya apa yang kita lakukan adalah sedang berinteraksi dengan object-object yang disebut Object-Oriented World. Contoh, saat kita ingin pergi ke sekolah kita membutuhkan kendaraan seperti mobil. Mobil merupakan sebuah object, dan mobil itu terbentuk dari object-object yang membentuknya, object yang membentuknya seperti roda, mesin, pintu dan lain-lain. Object yang membentuknya itu saling tersambung, berhubungan, berinteraksi, berkomunkasi, saling menyelesaikan tugasnya dan saling membutuhkan satu sama lain sehingga dapat membentuk satu object besar yaitu mobil, dan mobil bisa mengantar kita untuk pergi ke sekolah, selain itu object mobil juga bisa masuk kedalam object kendaraan atau transportasi.

<Gambar 1.1>

Dari pemikiran seperti diatas terciptalah sebuah pemikiran dalam pemrograman untuk membuat sebuah program dengan mudah, pemikiran tersebut dinamakan sebagai OOp Object Oriented programming. Dalam pemrograman, program besar dengan fungsi (Method) yang bermacam-macam dibentuk dari beberapa object yang lebih kecil dengan fungsi (Method) yang berbeda-beda, object-object tersebut saling tersambung, berhubungan dan saling berkomunikasi kepada object kecil lainnya, dan biasanya object kecil itu juga mempunyai object lainya dibawahnya (lebih kecil lagi).

Berdasarkan kutipan sejarah OOp diatas dan perkembangan konsep OOp yang sangat pesatdan hamir semua bahasa pemrgraman menggunakanya, maka konsep OOp ini dibagi menjadi 3, yaitu

Bahasa OOp Murni – sebuah bahasa yang mengharuskan program ditulis hanya berupa object saja. Contoh – Eifel, Smaltalk, Ruby, Jade dan lain-lain.

Bahasa OOp Hybrid – Bahasa yang dirancang untuk pemrograman object dengan beberapa elemen procedural.

Bahasa OOp Hybrid dalam Web – Sama seperti Bahasa OOp Hybrid, yang berbeda hanyalah, konsep ini sering digunakan dalam pemrograman Web.

C. Karakteristik OOP (Object Oriented Programming)

Semua adalah object.

Setiap object memiliki memori sendiri, yang biasanya dapat terdiri dari object-object lainnya.

Setiap object pada umumnya memiliki tiga sifat, yaitu keadaan, operasi dan identitas object.

Setiap object adalah wakil atau representasi dari suatu kelas. Sebuah kelas dapat mewakili sekelompok object yang sama.

Sebuah pesan merupakan permintaan atas sekumpulan aksi dengan semua argumen yang diperlukan untuk menyelesaikan suatu tugas tertentu.

Informasi yang terkandung pada object tersebut pada akhirnya memberikan identitas khusus yang membedakan suatu object dengan object lainnya.

Komputasi dilakukan dengan komunikasi antar object. Setiap object berkomunikasi dengan object yang lain melalui pengiriman dan penerimaan pesan.

Kelas merupakan kumpulan tingkah laku yang berkaitan dengan suatu object. Jadi, semua object yang merupakan wakil dari kelas yang sama dapat melakukan aksi yang

sama pula.

Operasi merupakan tindakan yang dapat dilakukan oleh sebuah object.

Keadaan object merupakan koleksi dari seluruh informasi yang dimiliki oleh object pada suatu saat.

Kelas-kelas diorganisasikan ke dalam struktur pohon yang berakar tunggal, yang dinamakan dengan jenjang pewarisan (inheritance hierarchy).

**Keuntungan Penggunaan Object Oriented Programming (OOP)**

1. Natural

OOp dapat membuat object pada pemrograman memiliki kegunaan, perilaku dan sifat yang nyata sebagai object yang seharusnya, atau biasanya object dalam pemrograman dbuat mirip seperti objek pada dunia nyata.

2. Modular

Object yang sudah dibentuk dapat dikelompokkan kembali dengan object-object yang lain, seperti kelompok alat tulis yang dapat dikelompokkan kembali dengan kelompok pensil, kelompok buku dan lain-lain.

3. Mudah diperbaharui

Dikarenakan sifat jangkauan dari object memiliki bagian private, protected dan public, maka jika bagian private ingin digunakan pada object-object lain dapat diperbaharui dengan menempatkan object lain tersebut di bagian public.

4. Dapat didaur ulang

Suatu object yang sudah didefinisikan dengan bentuk, sifat, perilaku, kegunaan, maupun jenis dapat didefinisikan kembali dengan object lain. Contoh, object mobil yang memiliki cirri umum bisa berjalan, mempunyai roda, mempunyai pintu dan lai-lain, dapat didefinisikan kembali ciri-ciri tersebut dengan menebut cirinya masing-masing seperti roda yang memiliki ciri-ciri seperti jenis, warna, ketebalan dan lain-lain.