Konsep OOP pada Pemrograman VB dan VB.Net

Sebelum memulai membangun sebuah sistem programmer hendaknya terlebih dahulu memahami konsep dasar pemrograman apakah termasuk Pemrograman terstruktur atau Pemrograman berorientasi objek.

*Pemrograman berorientasi Objek*yang dalam istilah Inggris disebut sebagai *Object Oriented Programming* (disingkat OOP) adalah salah satu pendekatan pemrograman atau paradigma untuk pengembangan / development suatu perangkat lunak komputer dimana dalam struktur perangkat lunak tersebut didasarkan kepada interaksi objek dalam penyelesaian suatu proses .

**OOP  dengan Procedural Programming**

Selain menggunakan paradigma OOP, programer dapat menuliskan struktur kode program yang baik dengan menggunakan *Procedural Programming*, di mana programer membuat kode program dengan pendekatan procedural dan bekerja dengan *function*/procedure di dalam aplikasinya. Tetapi, tidak ada jawaban yang benar-benar tepat jika Anda diberi pertanyaan: apakah sebaiknya menggunakan OOP atau Procedural Programming? Karena jawabannya sangat relatif, terutama tergantung pada aplikasi yang ingin Anda buat. Jika Anda mempertimbangkan pemeliharaan dan pengembangan aplikasi yang eﬁsien di masa yang akan datang, mungkin Anda dapat memilih pendekatan OOP. Tetapi, jika aplikasi Anda merupakan program sederhana yang dapat dibuat dengan mudah dan cepat dengan function/procedure, gunakanlah pendekatan Procedural Programming. Semua itu juga tergantung pada Anda sebagai programer, untuk memilih pendekatan yang cocok dan lebih baik bagi Anda dalam mengerjakan sebuah aplikasi.

**Apakah VB6 menggunakan Konsep OOP ?**

Pada Visual Basic 6.0 (dan Visual Basic versi sebelumnya), banyak diperdebatkan apakah bahasa pemrograman Visual Basic termasuk sebagai bahasa pemrograman berorientasi object ataukah tidak. Pada satu sisi, Visual Basic 6.0**mendukung** object dan interface, dan juga mengimplementasikan class yang dideﬁnisikan pada class module dan disimpan pada ﬁle dengan ekstensi .cls. Bahkan implementasi *drag and drop control* yang pasti dikenal oleh setiap programer Visual Basic merupakan contoh object yang dapat di-*reuse.*Pada saat Anda meletakkan sebuah control, misalnya sebuah *command button* pada form Visual Basic, Anda telah menciptakan object baru yang memiliki members berupa *property, method,* dan *events*. Visual Basic 6.0 juga mengenal *keyword private, friend, public,* dan *static* yang digunakan untuk mengatur level akses dari elemen yang dideklarasikan. Tetapi, pada sisi lain Visual Basic 6.0 dianggap **tidak mengimplementasikan inheritance** dan hanya memiliki ﬁtur object oriented yang terbatas. Sebuah class pada Visual Basic 6.0 tidak mendukung untuk menurunkan property dan function-nya pada class yang lain. Bagaimana dengan **polymorphism**? Visual Basic 6.0 **mendukung polymorphism melalui interface pada ActiveX**. Pada *Component Object Model* (COM) yang membentuk infrastruktur spesiﬁ kasi ActiveX, terdapat *multiple interface* yang memungkinkan komponen software dikembangkan tanpa mengubah kode program yang ada. Secara umum, OOP pada Visual Basic 6.0 lebih diimplementasikan pada ***user interface***dan kurang memberikan dukungan bagi programer untuk mengimplementasikan OOP pada kode program. Paradigma ini yang kemudian diubah pada **generasi Visual Basic .NET**.

**Bagaimana dengan VB.Net ?**

Perdebatan apakah Visual Basic merupakan bahasa pemrograman berorientasi object ataukah tidak, berakhir pada saat dirilisnya Visual Studio .NET dan generasi seterusnya. Pada .NET, Visual Basic murni merupakan bahasa pemrograman berorientasi object. Kegunaan OOP yang utama adalah memaksimalkan penggunaan kembali kode program (*code reuse*). Saat Anda melakukan drag and drop, prosesnya dikerjakan oleh *control designer* sehingga Anda tidak perlu mengetahui bagaimana object itu bekerja. Bahkan Anda bebas menggunakan sebanyak mungkin control dan memodiﬁkasi warna, bentuk, judul, bahkan event handling dari object. Dasar dari tipe sistem .NET adalah inheritance, yang tidak Anda temui dalam Visual Basic sebelumnya. Contoh implementasi inheritance pada .NET adalah *namespace*, .NET mengenal enam tipe kategori yang dapat dideﬁnisikan pada namespace:

**1. Classes.**

Merupakan tipe referensi yang dideﬁ nisikan dengan struktur Class .. End Class.

**2. Arrays.**

Merupakan tipe referensi yang menyimpan object dari tipe yang lain. Class Array didefinisikan pada namespace System pada .NET Framework Class Library.

**3. Structures.**

Dideﬁnisikan dengan struktur Structure … End Structure.

**4. Interfaces.**

Dideﬁnisikan dengan struktur Interface .. End Interface.

**5. Delegates.**

Merupakan tipe referensi yang mengenkapsulasi method dan didefinisikan dengan *statement Delegate.*

**6. Enumerations.**

Merupakan kumpulan nilai yang berkaitan, dideﬁnisikan dengan struktur Enum ..End Enum.

Masing-masing dari enam kategori tersebut dapat mendeﬁnisikan satu atau lebih members. Tipe-tipe members pada .NET adalah:

**1. Fields.**

Merupakan konstanta atau variabel yang mengizinkan akses tipe data. Fields dapat dideﬁ nisikan oleh *class, structure,* dan *enumeration*. Fields digunakan sebagai data yang bersifat *read-only,* atau sering dideﬁnisikan sebagai konstanta. Fields juga dapat berupa variabel read-only yang nilainya dideﬁnisikan kali pertama pada class constructor dan tidak dapat diubah lagi.

**2. Properties.**

Property dapat dideﬁ nisikan untuk class, structure, dan interface. Umumnya property dapat dibaca (*readable*) dan ditulis (*writable*), walaupun dapat juga diperlakukan read-only atau write-only.

**3. Methods.**

Merupakan function atau subrutin yang tersedia pada sebuah class, structure, interface, atau delegate. Pada sebuah method terdapat parameter atau argumen yang dapat dikirimkan *by value* atau *by reference*. Bagi programer Visual Basic 6.0 sudah tentu tidak asing lagi dengan parameter by value maupun by reference yang biasa digunakan dalam pembuatan function/procedure. By value berarti perubahan nilai pada variabel parameter yang dikirimkan hanya akan berlaku di dalam function, sementara by reference berarti perubahan nilai pada variabel di dalam function akan tetap berlaku pada saat function tersebut selesai dieksekusi dan kembali pada pemanggil. Dengan menggunakan keyword ByVal atau ByRef, Anda dapat menentukan bagaimana parameter dikirimkan.

**4. Events.**

Pada Visual Basic versi sebelumnya, Anda tentu telah mengenal event yang menimbulkan istilah *event-driven programming*. Event merupakan sebuah function yang dipanggil untuk merespon kejadian/event tertentu, seperti klik mouse, penekanan tombol keyboard, perubahan nilai ﬁeld pada database, dan lain-lain. Umumnya event mengirimkan dua parameter, yaitu object yang mengindikasikan sender/pengirim event, dan object yang menyediakan informasi pada event tersebut.

Pada sebuah class, Anda dapat melakukan modiﬁkasi pada method dengan cara*overriding* maupun *overloading*. Overriding berarti Anda mengubah implementasi dari sebuah method, sementara overloading berarti menambahkan method baru dengan nama yang sama, tetapi dengan pengenal yang berbeda. Overloading mengizinkan Anda memiliki dua atau lebih method dengan nama yang sama. Dua method tersebut akan muncul pada fasilitas IntelliSense sehingga pengguna dapat memilih method yang diinginkan. Bagaimanapun, salah memilih method dapat menyebabkan *runtime-error,* Anda dapat menghilangkan method yang tidak diinginkan untuk tampil dengan menggunakan keyword Shadows.

**Generic Class**

Inovasi baru generasi .NET adalah ﬁtur yang dikenal dengan *generics* yang menyediakan sejumlah *generic class*. Untuk memahami kegunaan dari generic class, bayangkan Anda ingin membuat berbagai versi sebuah class di mana Anda ingin class tersebut bekerja pada lebih dari satu tipe data, misalnya class yang dapat bekerja pada tipe data integer, string, dan lain-lain. Dengan generic class, Anda tidak perlu menentukan tipe data yang diinginkan hingga Anda menciptakan object-nya di dalam program Anda. Anda dapat menemukan generic class pada namespace System. Collections.Generic pada .NET Framework Class Library. Selain menggunakan generic class yang tersedia pada .NET Framework, Anda juga dapat menuliskan sendiri generic class Anda. Dapat disimpulkan keuntungan dari generic class adalah:

**1. Keamanan tipe data.**

Generic class memeriksa kevalidan tipe data pada saat *compile-time* dan bukan pada saat *run-time*. Hal ini mengurangi kemungkinan konﬂ ik tipe data pada saat run-time.

**2. Kinerja yang lebih baik.**

Karena tipe data yang digunakan pada generic class dikenali pada saat compile-time, maka tidak diperlukan proses type casting pada saat run-time yang tentunya mempercepat waktu proses.

**3. Penggunaan ulang (reuse).**

Anda hanya perlu menuliskan class satu kali dan dapat menggunakannya dengan tipe data yang bervariasi.

**Partial Class**

Pada .NET versi 2.0, dikembangkan partial class atau class yang terpisah, yang memungkinkan Anda untuk memisahkan deﬁ nisi class pada beberapa ﬁle. Pada saat compile-time, compiler akan mengelompokkan partial class tersebut dan memperlakukannya sebagai satu kesatuan. Keuntungan dari partial class adalah mengelompokkan pada batas yang jelas antara *business logic* dan user interface. Dengan partial class, kode program yang merupakan user interface tidak perlu ditampilkan pada programer, yang pada umumnya tidak perlu mengaksesnya.Partial class juga mempermudah proses *debugging*, di mana kode dipartisi ke dalam beberapa ﬁle yang berbeda.

**Kesimpulan :**

Inti dari Konsep OOP adalah memiliki Sifat Enkapsulasi,Inheritence (Turunan),Polimorpisme (Banyak bentuk) Sedangkan Pemrograman terstruktur untuk inheritance dan polimorfisme tidak mampu melakukan secara murni kecuali dengan interface.  
Itu yang menyebabkan beberapa kalangan menyebut VB6 bukan bahasa ber-orientasi objek murni

**Saran :**

Tidak semua Aplikasi yang dibangun harus dengan konsep OOP tetapi untuk Aplikasi yang memungkinkan akan terjadi pengembangan bahkan sampai Aplikasi yang berskala besar seharusnya menggunakan konsep OOP agar *maintenance(Perawatan)*menjadi lebih mudah

<https://mangmadi.wordpress.com/2009/12/28/penerapan-konsep-oop-pada-pemrograman-vb-dan-vb-net/>