
MODUL IV

PEMROGRAMAN BERORIENTASI OBJEK

A. TUJUAN

- ✓ Memahami konsep dasar pemrograman berorientasi objek.
- ✓ Mampu mengimplementasikan konsep-konsep pemrograman berorientasi objek di dalam program.
- ✓ Mampu menyelesaikan kasus-kasus sederhana dengan menggunakan paradigma objek.

B. PETUNJUK

- Awali setiap aktivitas dengan doa, semoga berkah dan mendapat kemudahan.
- Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas praktikum dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas.

C. DASAR TEORI

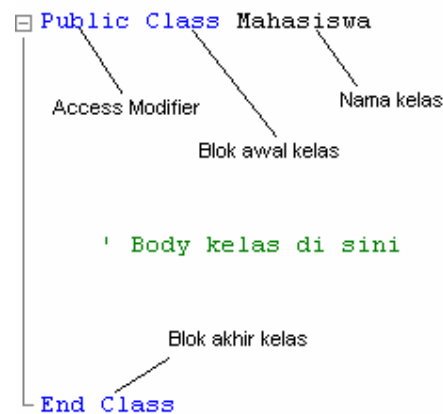
1. Pemrograman Berorientasi Objek

Secara garis besar, suatu bahasa pemrograman dapat dikatakan sebagai bahasa pemrograman berorientasi objek (atau *Object Oriented Programming / OOP*) apabila ia mendukung konsep abstraksi (*abstraction*), enkapsulasi (*encapsulation*), polimorfisme (*polymorphism*), dan pewarisan (*inheritance*). Selain konsep-konsep ini, ada beberapa konsep fundamental lainnya, seperti kelas, objek, dan *message*.

2. Kelas

Kelas mendefinisikan karakteristik-karakteristik abstrak dari sesuatu (objek), termasuk karakteristik dan perilaku (*behavior*) dari “sesuatu” itu sendiri. Kelas dapat diilustrasikan sebagai sebuah cetak biru (*blueprint*), prototipe, atau pabrik (*factory*) yang berfungsi untuk menghasilkan objek-objek.

Bentuk kelas yang paling sederhana diperlihatkan sebagai berikut:



3. Objek

Dalam terminologi OOP, objek adalah instans (atau manifestasi) dari sebuah kelas. Dengan demikian, dalam konteks desain, kita berbicara mengenai kelas; saat *run time*, yang kita bicarakan adalah objek.

Baik di dunia nyata maupun di dalam pemrograman, sebuah objek memiliki dua karakteristik utama, yaitu *state* (status) dan *behavior* (perilaku). Sebagai contoh, kucing memiliki *state* (nama, warna, dan sebagainya) dan *behavior* (mengeong, melompat, dan sebagainya).

4. Field

Field adalah variabel yang didefinisikan di dalam kelas, dan disebut juga sebagai member variable. Field—dan juga member-member kelas lainnya—dapat dideklarasikan dengan level akses tertentu. Berkaitan dengan level akses ini, ada beberapa jenis level dari yang umum sampai yang restriktif.

Access Modifier	Keterangan
Public	Untuk mendefinisikan tipe yang bisa diakses oleh siapa saja.
Friend	Untuk mendefinisikan tipe yang hanya bisa diakses dari <i>current</i> project, atau dari assembly di mana tipe tersebut dideklarasikan.
Protected	Mendefinisikan tipe yang hanya bisa diakses oleh member-member kelas itu sendiri atau member kelas turunan.
Protected Friend	Untuk mendefinisikan tipe yang bisa diakses oleh member-member dalam satu assembly atau kelas turunannya.
Private	Mendefinisikan tipe yang hanya bisa diakses oleh member-member di mana tipe tersebut dideklarasikan.

D. LATIHAN

1. Kelas dan Objek

Sebelum mendefinisikan kelas, terlebih dahulu kita menciptakan project Visual Basic.

1. Buat aplikasi Windows (dengan memilih template **Windows Application**)
2. Tambahkan item kelas baru melalui menu **Project > Add Class**. Simpan dengan nama **Mahasiswa.vb**.
3. Definisikan konstruktor dan properti pada kelas Mahasiswa.

```
Public Class Mahasiswa
    ' Field nim dan nama
    Private mNim As Integer
    Private mName As String

    ' Konstruktor
    Sub New(ByVal mNim As Integer, ByVal mName As String)
        Me.mNim = mNim
        Me.mName = mName

        ' Sekadar info
        Console.WriteLine("Konstruktor dipanggil")
    End Sub

    ' Properti Nim (setter/getter)
    Public Property Nim() As Integer
        Get
            Return mNim
        End Get
        Set(ByVal value As Integer)
            mNim = value
        End Set
    End Property

    ' Properti Nama (setter/getter)
    Public Property Nama() As String
        Get
            Return mName
        End Get
        Set(ByVal value As String)
            mName = value
        End Set
    End Property
End Class
```

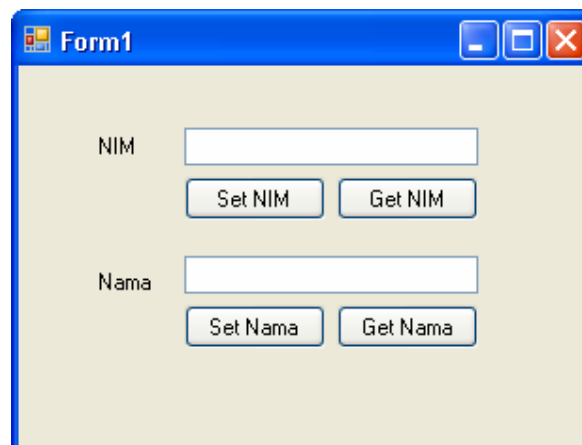
4. Simpan kelas Mahasiswa.

Setelah kelas terdefinisi, kita dapat menggunakannya sebagaimana tipe—karena pada hakekatnya ia merupakan tipe referensi. Sebagai contoh, kita memanfaatkan aplikasi Windows untuk menguji fungsionalitas objek Mahasiswa.

1. Masih di project yang sama, tampilkan desain form.
2. Tambahkan kontrol-kontrol dengan spesifikasi sebagai berikut:

Kontrol	Properti	Nilai
Label	Name	Label1
	Text	NIM
TextBox	Name	txtNIM
	Text	
Button	Name	btnSetNIM
	Text	Set NIM
Button	Name	btnGetNIM
	Text	Get NIM
Label	Name	Label2
	Text	Nama
TextBox	Name	txtNama
	Text	
Button	Name	btnSetNama
	Text	Set Nama
Button	Name	btnGetNama
	Text	Get Nama

3. Bentuk desain form-nya misalkan terlihat seperti berikut:



Gambar 1 Desain form

4. Deklarasikan field mahasiswa, kemudian tambahkan event Load pada form untuk menciptakan objek Mahasiswa.

```
' Deklarasi field mahasiswa
Private mhs As Mahasiswa

Private Sub Form1_Load(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.Load
    ' Menciptakan objek Mahasiswa
    mhs = New Mahasiswa(1, "Agus")
End Sub
```

5. Tambahkan event Click pada button btnSetNIM, kemudian lengkapi kode event-handler-nya.

```
Private Sub btnSetNIM_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnSetNIM.Click
    ' Parse string ke integer
    ' Bisa juga dengan CInt, tapi lebih disukai cara ini
    mhs.Nim = Integer.Parse(Me.txtNIM.Text)
End Sub
```

6. Tambahkan event Click pada button btnGetNIM, kemudian lengkapi kode event-handler-nya.

```
Private Sub btnGetNIM_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnGetNIM.Click
    MessageBox.Show("NIM: " & mhs.Nim)
End Sub
```

7. Tambahkan event Click pada button btnSetNama, kemudian lengkapi kode event-handler-nya.

```
Private Sub btnSetNama_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnSetNama.Click
    ' Men-set nama mahasiswa
    mhs>Nama = Me.txtNama.Text
End Sub
```

8. Tambahkan event Click pada button btnGetNama, kemudian lengkapi kode event-handler-nya.

```
Private Sub btnGetNama_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnGetNama.Click
    MessageBox.Show("Nama: " & mhs>Nama)
End Sub
```

9. Jalankan aplikasi, uji, dan pahami hasil keluarannya.

2. Method

Method merepresentasikan aksi suatu objek yang dapat dipanggil (*di-*invoke**), dan didefinisikan melalui prosedur atau fungsi. Dengan kata lain, method pada dasarnya adalah prosedur atau fungsi; jadi, istilah method akan mengacu pada keduanya.

```
' Contoh method, misal hanya sekedar mengembalikan string
' dalam format huruf besar
Public Function UpperName(ByVal str As String) As String
    Return str.ToUpper()
End Function
```

Di Visual Basic, kita juga bisa mendefinisikan method statis dengan menggunakan keyword `Shared`.

```
' Contoh method statis
Public Shared Function UpperName(ByVal str As String) As String
    Return str.ToUpper()
End Function
```

Seperti di kebanyakan bahasa pemrograman, method statis *di-*invoke** melalui nama kelasnya (bukan instans kelas).

```
NamaKelas>NamaMethod()
```

3. Enkapsulasi

Enkapsulasi (encapsulation), atau lazim disebut pembungkusan, merupakan sebuah bentuk privasi yang diterapkan pada member-member kelas. Enkapsulasi dapat diartikan sebagai pembungkusan data (atribut atau properti) dan method ke dalam sebuah unit tunggal—yang disebut objek.

Perhatikan contoh kasus berikut:

```
Public Class NonEnkapsulasi

    Public strNama As String
    Public strAlamat As String

    Public Sub GetInfo()
        Console.WriteLine("Nama: " & Me.strNama)
        Console.WriteLine("Alamat: " & Me.strAlamat)
    End Sub

End Class
```

Oleh karena `strNama` dan `strAlamat` dideklarasikan `Public`, maka ia bisa diakses dari mana saja.

```
Dim ne As New NonEnkapsulasi

' Men-set nama dan alamat
ne.strNama = "nama"
ne.strAlamat = "alamat"

' Mendapatkan info
ne.GetInfo()
```

Contoh di atas memperlihatkan kasus yang melanggar konsep enkapsulasi. Adapun solusinya, deklarasikan field dengan access modifier `Private`, kemudian gunakan setter/getter (properti) untuk mengakses field.

E. TUGAS PRAKTIKUM

1. Buat kelas `Day` dengan sebuah method statis bernama `GetDay()`. Definisikan juga kelas untuk menguji fungsionalitas kelas `Day`.

Petunjuk:

Gunakan properti `Now` untuk mendapatkan *current day*.

2. Buat kelas `Point` dengan atribut `x` dan `y`, kemudian uji fungsionalitasnya dengan mendefinisikan kelas lain, misalnya `PointDemo`.
3. Definisikan kelas `Circle` dengan atribut jari-jari dan `pi`, serta operasi `GetArea()`.

Petunjuk:

Gunakan keyword `Const` untuk mendefinisikan konstanta `PI`.