SOETAM RIZKY

LEARNING BY SAMPLE

VISUAL BASIC 2008

Jakarta 2010
PRESTASI PUSTAKARAYA
Penerbit Buku Berkualitas Prima

SOETAM RIZKY
Learning by Sample
VISUAL BASIC 2008

Editor: **Sofan Amri**
Desain Cover: **Sudarmaji Lamiran**
Setting: **Tim Prestasi**

E-mail: redaksi@prestasipustakaraya.com

SOETAM RIZKY
Learning by Sample
VISUAL BASIC 2008

ISBN: 978-602-8470-64-3

Cetakan Pertama: Maret 2010

Learning by sample – VB 2008 Jilid 1

Untuk istri dan putri kecilku,
pelipur jiwa di kala duka…..

Kekayaan adalah saat semua yang dimiliki
mampu mencukupi segala kebutuhan kita,
Kemiskinan adalah saat semua yang dimiliki
tak mampu mencukupi segala kebutuhan kita

Kata Pengantar

Alhamdulillah, akhirnya buku yang merupakan bagian pertama dari dua jilid yang sudah direncanakan. Buku ini merupakan buku kedua dari seri Learning By Sample, setelah sebelumnya menyelesaikan seri Learning By Sample untuk ASP .NET 3.5.

Tiada karya yang sempurna, begitu pula yang terjadi di buku ini. Dengan berbagai kekurangannya, penulis tetap berharap agar semua yang tertera di buku ini dapat bermanfaat bagi seluruh pembacanya.

Ucapan terima kasih yang dalam kepada pihak yang telah membantu secara langsung maupun tidak langsung dalam proses terbitnya buku ini. Kepada Pak David Oscar yang tak pernah bosan memberikan dukungannya, kepada rekan-rekan di Universitas Ma Chung, dan juga sahabat lama di STIKOM Surabaya yang secara moril tetap membuat semangat menulis menyala tanpa henti. Juga kepada para pembaca yang rajin dan setia

mengirimkan email dan juga komentar di facebook berupa saran dan kritikan.

Semoga apa yang tersurat dan tersirat dalam buku ini dapat menjadi sebuah sumber ilmu dan inspirasi bagi kita semua untuk tetap belajar tanpa batas. Selamat berkarya……

Malang, Akhir Desember 2009

Soetam Rizky

Daftar Isi

Pendahuluan

Mengapa Learning By Sample ?

Setelah lebih dari lima tahun mengajarkan (dan juga belajar) mengenai pemrograman visual baik berbasis desktop maupun berbasis web, terdapat satu pelajaran penting yang diambil dari sebuah proses pembelajaran. Bahwa proses belajar pemrograman sangatlah sulit dilakukan jika diawali dengan sebuah teori dan konsep yang terlalu *tinggi*.

Hal tersebut akan mengakibatkan para pemula patah arang sebelum memulai proses belajar dengan langsung membayangkan betapa sulitnya belajar sebuah bahasa pemrograman. Tentu saja hal tersebut sangatlah merugikan bagi para pembelajar, baik yang ada di dunia akademik maupun dunia profesional.

Dan setelah mencoba dengan berbagai metode, muncul sebuah kesimpulan bahwa ternyata proses belajar yang dianggap penulis hingga saat ini paling efektif dan efisien adalah belajar pemrograman dengan cara langsung mencoba membuat aplikasi-aplikasi dari yang

level paling sederhana hingga level kompleks secara bertahap. Metode yang juga sering disebut dengan *Learning by Sample* tersebut memang sudah terbukti ampuh untuk mempelajari sebuah bahasa pemrograman lebih cepat dan lebih baik.

Metode tersebut sesungguhnya telah diterapkan pula oleh Microsoft sendiri dengan menerbitkan seri *HOL (Hands On Lab)* yang menampilkan contoh-contoh pemrograman dengan level tertentu dan langsung dipraktekkan oleh pengguna. Sayangnya, HOL yang berbahasa Inggris dan jarang diketahui oleh para programmer pemula masih tidak populer penggunaannya baik di kalangan akademisi maupun profesional.

Buku ini memang tidak bermaksud untuk menggantikan HOL ataupun melakukan duplikasi contoh-contoh yang ada di dalam HOL. Sebab seluruh aplikasi yang ada di dalam buku ini dijamin orisinalitasnya dan benar-benar hasil *hand-made* dari penulis berdasarkan contoh yang

sudah diterapkan dalam proses belajar mengajar baik di kelas maupun di lab.

Tetapi meski pada bagian awal hanya menampilkan contoh aplikasi (lengkap dengan penjelasan cara pembuatannya), bukan berarti buku ini menafikan teori dari pembelajaran bahasa pemrograman itu sendiri. Karenanya di bagian akhir, terdapat appendix atau tambahan yang menjelaskan secara ringkas mengenai konsep yang diterapkan di dalam contoh-contoh sebelumnya.

Diharapkan pula, bagi Anda para pemula, dapat mencoba satu per satu contoh dalam buku ini yang memang sengaja dibuat *lepas* satu sama lain, sehingga proses belajar pun dapat dilakukan dengan cara loncat bab sesuai dengan kebutuhan masing-masing pembaca. Dan sangat diharapkan pula, bahwa contoh-contoh sederhana yang ada dalam buku ini tidak hanya berhenti saat Anda telah selesai mengerjakannya. Karenanya di tiap akhir contoh terdapat tip kecil saran pengembangan yang dapat digunakan sebagai

sarana untuk melakukan pengembangan diri dan mengasah logika pemrograman lebih lanjut.

Apa Yang Ada Dalam Buku Ini ?

Di dalam buku ini terbagi menjadi dua bagian utama yakni bagian contoh aplikasi dan bagian appendix. Khusus di bagian contoh aplikasi terbagi lagi menjadi empat bab yakni bab pertama atau dasar pemrograman visual yang membahas contoh-contoh yang dikategorikan sebagai contoh paling dasar bagi para pemula. Di dalam bab pertama dapat dipelajari teknik dasar pemrograman visual, penempatan komponen sekaligus pengenalan mengenai logika pemrograman sederhana yakni perulangan dan percabangan.

Sedangkan bab kedua dari bagian contoh aplikasi membahas pemrograman visual lanjutan yang didalamnya mencoba mengaplikasikan teknik lanjutan dari komponen visual dasar yang ada di dalam Visual Basic 2008. Beberapa komponen yang dibahas didalamnya antara lain mengenai penggunaan Timer, Rich Text Editor dan juga Picture Box. Selain itu, di bab ini juga dibahas mengenai teknik lanjutan pemrograman seperti

pembangkitan bilangan acak, akses file teks secara sekuensial serta manipulasi file secara umum dengan menggunakan komponen jenis kotak dialog.

Di bab ketiga dari bagian contoh dibahas mengenai pengenalan implementasi konsep OOP atau Object Oriented Programming. Selain membahas contoh dasar implementasi class dan proses inheritance, juga di contoh lainnya dibahas mengenai pembuatan user control baik yang bertipe single maupun composite. Dalam contoh-contoh tersebut, juga dijelaskan mengenai pembuatan dan penggunaan property serta implementasi dari teori polymorphisme di dalam konsep OOP.

Di bab terakhir bagian contoh, dijelaskan mengenai aplikasi pengembangan yang didalamnya memiliki contoh mengenai implementasi GDI+, proses enkripsi serta akses file XML. Dalam contoh di bab ini, meskipun terlihat lebih pendek dibanding contoh pada bab sebelumnya tetapi dikategorikan sebagai contoh

bagi para pemula yang telah mencapai level mahir. Tetapi bukan berarti contoh dalam bab ini mustahil untuk dilakukan oleh para pemula, karena meski konsep yang diusung memiliki level mahir tetapi contoh yang disajikan tetap berpegang pada prinsip kesederhanaan.

Di semua bab yang mengandung contoh aplikasi, selalu dilengkapi dengan keterangan listing serta tip-tip yang ditandai dengan icon tanda seru disampingnya. Sehingga diharapkan, pembaca tidak hanya sekedar melakukan pengetikan listing dan perancangan form, tetapi juga mampu memahami aplikasi yang dibuat dengan model pemikiran yang sederhana. Dengan konsep sederhana pula, diharapkan bahwa buku ini sanggup menjadi teman belajar secara mandiri bagi para pemula, tetapi juga tidak menutup kemungkinan untuk diimplementasikan bagi para rekan-rekan di dunia akademik untuk membawa rangkaian contoh ini ke dalam aktifitas kelas atau lab.

Di bagian kedua atau bagian appendix, memang secara umum membahas sekilas mengenai teori yang ada di dalam contoh-contoh yang ditampilkan di bab-bab sebelumnya. Meski demikian, bukan berarti bagian appendix adalah bagian yang mengandung teori *high class*. Sebab dengan tujuan utama buku ini yang lebih mengarah ke para pemula, maka teori yang ada dikemas dengan bahasa yang lebih sederhana dan diharapkan tidak terlalu banyak menimbulkan bias dalam menerapkan konsep *learning by sample* yang dianut oleh buku ini.

Apa Yang Dibutuhkan ?

Bagi Anda para pemula, seringkali bertanya sebelum memulai belajar bahasa pemrograman baru. Apa yang harus saya sediakan atau apa yang dibutuhkan sebelum memulai belajar bahasa pemrograman tersebut ?

Seluruh contoh yang ada di dalam buku ini pada awalnya dibuat dengan menggunakan Visual Studio 2005 Express Edition, tetapi pada saat proses penulisan, dengan diluncurkannya Visual Studio 2008, maka seluruh aplikasi di dalam buku ini dikonversi ke versi 2008. Seluruh aplikasi contoh pada dasarnya dibuat sepenuhnya dengan menggunakan Visual Studio 2008 Express Edition Service Pack 1, tetapi juga dites ulang dengan menggunakan Visual Studio 2008 Profesional Edition.

Penggunaan Visual Studio 2008 Express Edition, selain gratis (benar-benar gratis meski tidak open source) juga mudah dalam proses instalasi. Selain itu juga ukurannya yang kecil dan tidak terlalu besar dalam kebutuhan sumber daya

perangkat keras menjadikan versi tersebut sangat sesuai untuk para pemula. Meski demikian, versi Express juga dapat digunakan untuk aplikasi profesional meski memiliki beberapa keterbatasan didalamnya.

Ini berarti bahwa yang Anda butuhkan sebelum memulai membaca dan juga mempraktekkan buku ini adalah :

1. Visual Studio 2008 Express Edition SP1 (dapat didownload secara gratis di situs Microsoft).
2. .NET Framework 3.5 (terinstalasi secara otomatis di Windows Vista, dan akan terinstalasi otomatis saat instalasi Visual Studio)
3. Secangkir kopi motivasi dan sepotong roti kemauan untuk belajar pantang menyerah.

Selamat berkarya !!!!

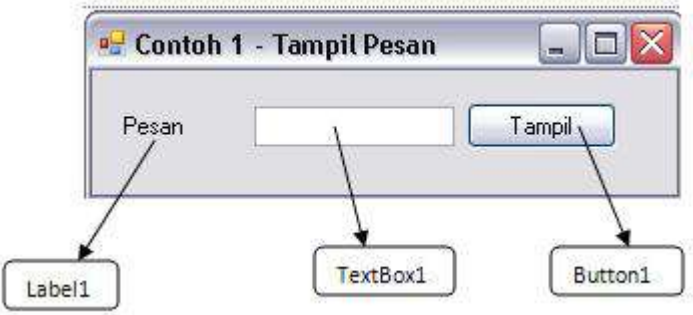
Dasar Pemrograman Visual

Contoh 1 : Tampil Pesan

Level : Pemula
Tujuan :

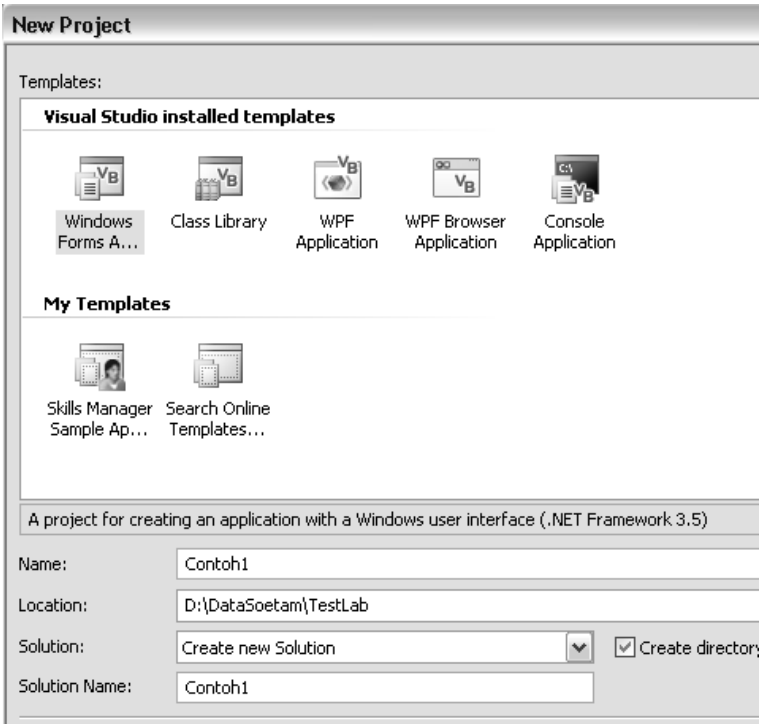
1. Mengenal pembuatan solution
2. Mengenal cara *resize* form
3. Mengenal setting form pada saat *design time*
4. Membuat form untuk menampilkan pesan berdasarkan isian pengguna.
5. Mengenal Textbox, Button dan Messagebox
6. Mengenal setting property komponen dasar pada saat *design time* (Textbox, Label dan Button).

Desain form awal :



Langkah penyelesaian :

1. Buat sebuah solution baru dengan nama *Contoh1* di dalam lingkup Visual Basic Express Edition (dengan tampilan yang sedikit berbeda jika Anda menggunakan Visual Studio 2008).

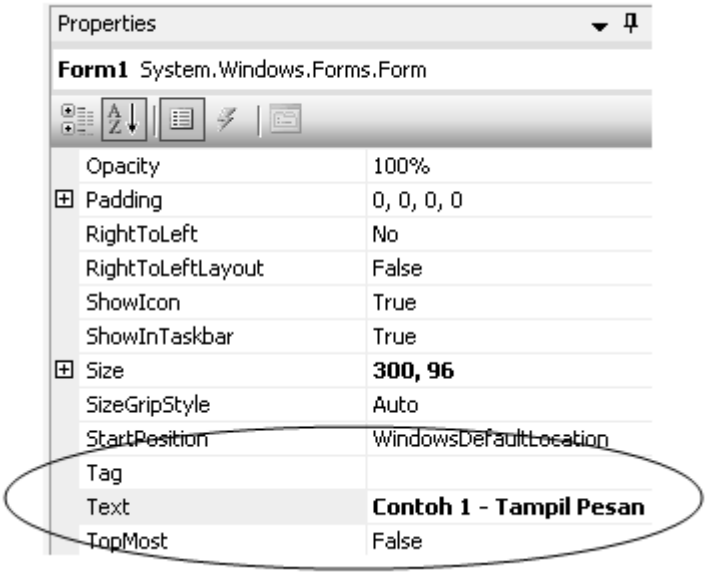


Visual Basic 2008 secara default menggunakan *folder based solution* yang berarti akan membuat sebuah folder baru pada setiap solution baru yang dibuat. Nama folder baru akan disesuaikan dengan nama solution yang dibuat.

2. Di dalam form yang tersedia, perkecil ukuran dengan menggunakan mouse (dengan melakukan drag pada ujung kiri bawah dari

form sehingga ukuran yang tersedia hampir sama dengan desain form awal).

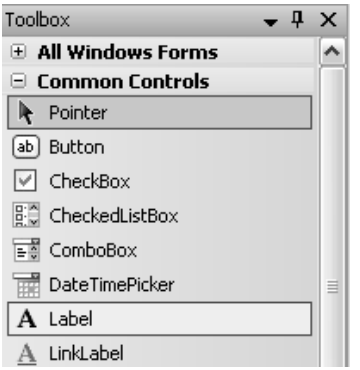
3. Di dalam jendela *Properties*, pilih property *Text* untuk form dan set nilainya menjadi *Contoh 1 – Tampil Pesan*



Setting property pada saat aplikasi belum dieksekusi dinamakan sebagai setting property pada saat *design time*. Sedangkan setting property yang dilakukan di dalam listing program dan efeknya bisa dilihat pada saat aplikasi dijalankan dinamakan setting property pada saat *run time*.

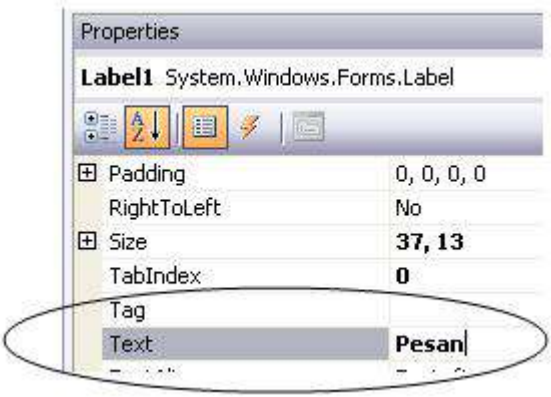
Pada setting property *Text* di dalam form , dapat dilihat langsung hasilnya di *caption* (judul form) pada saat itu juga.

4. Berikutnya, drag sebuah komponen *Label* dari Toolbox ke dalam form.

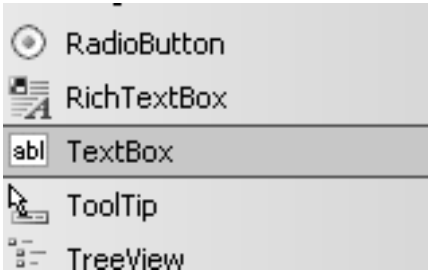


Komponen Label umumnya hanya digunakan untuk keterangan dalam sebuah form.

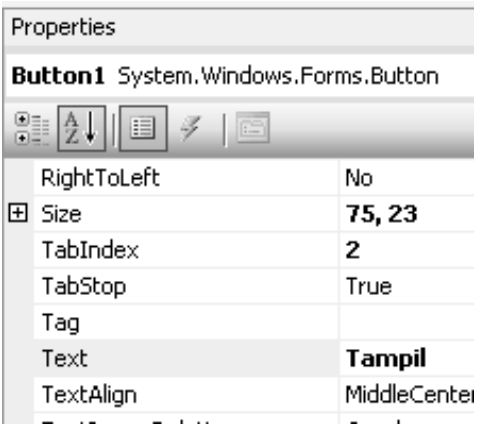
5. Pada komponen Label tersebut, set property *Text* dengan isi nilai *Pesan*. Lalu atur posisi sedemikian rupa sehingga mendekati gambar desain form awal.



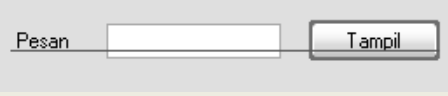
6. Selanjutnya, drag komponen *Textbox*, dari Toolbox tepat di sebelah kanan komponen Label, dan atur sedemikian rupa lebarnya sehingga mendekati desain form awal.



7. Kemudian drag komponen *Button*, tepat di sebelah kanan komponen Textbox. Lalu set property *Text* pada komponen tersebut menjadi *Tampil*.



Agar ketiga komponen tersebut bisa lurus, manfaatkan fasilitas *ruler* yang ada di dalam Visual Basic 2008, dengan melakukan drag pada salah satu komponen, maka akan terlihat sebuah (atau lebih) garis yang menandakan lurus tidaknya posisi ketiga komponen tersebut.



8. Berikutnya, dobel klik pada button *Tampil* sehingga tampilan akan menuju ke tampilan mode listing. Dan prosedur standar untuk button adalah menangkap kejadian (event) pada saat button diklik oleh pengguna.

9. Di dalam prosedur tersebut, ketikkan listing berikut ini :

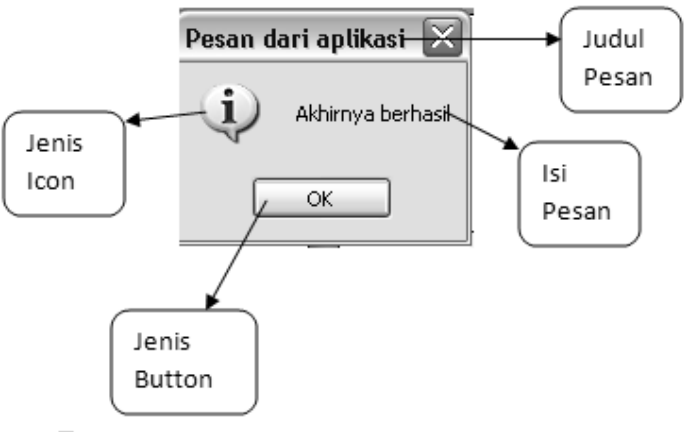
```
MessageBox.Show(TextBox1.Text, _
 "Pesan dari aplikasi", MessageBoxButtons.OK, _
            MessageBoxIcon.Information)
```

10. Dan untuk menjalankan aplikasi pertama ini, tekan icon *Start Debugging*  atau sama dengan menekan tombol F5.

Keterangan Listing :
Perintah *MessageBox* digunakan untuk menampilkan kotak pesan dengan isi teks tertentu. Sintaks sederhana dari perintah tersebut adalah :
MessageBox.Show(isi pesan, judul pesan, jenis button yang ditampilkan, jenis icon)

Hasil akhir :

Setelah textbox diisi dan diklik button pesan, maka akan muncul kotak pesan sesuai dengan isi textbox tersebut :

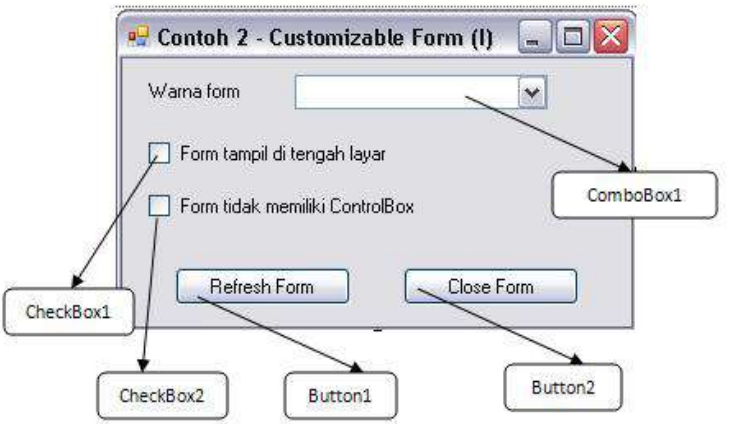


Saran Pengembangan :
Cobalah untuk menampilkan perintah *MessageBox* dengan menggunakan parameter lain, misalnya dengan menggunakan *multiple icon* seperti OK-Cancel, Yes-No dan lainnya. Cobalah pula untuk menampilkan paramter judul pesan dari textbox baru di dalam program sederhana tersebut.

Contoh 2 : Customizable Form

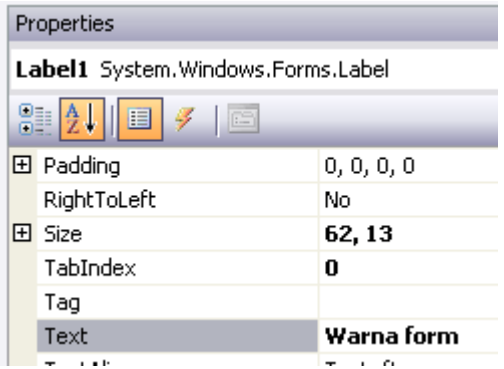
Level : Pemula
Tujuan :
1. Membuat form yang dapat diatur tampilannya melalui setting property runtime
2. Mengenalkan property form
3. Mengenalkan komponen Checkbox dan logika percabangan
4. Mengenalkan komponen Combobox

Desain form awal :

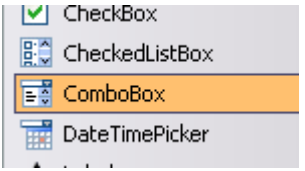


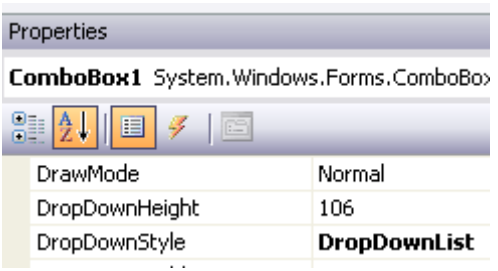
Langkah penyelesaian :

1. Buat sebuah solution baru dengan nama *Contoh2*
2. Di dalam form yang tersedia, drag komponen-komponen berikut ini ke dalam form :
    a. Sebuah komponen label dan set property *Text* menjadi *Warna Form*



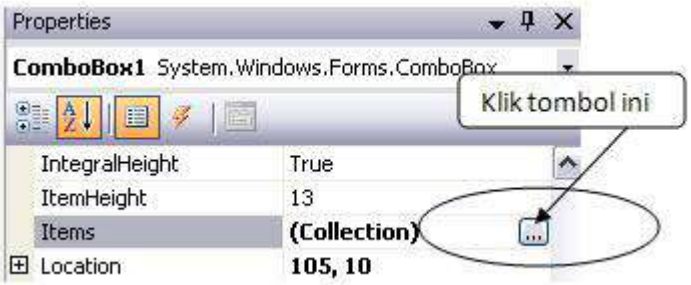
b. Sebuah komponen *Combobox* dan set property *DropDownStyle* menjadi *DropDownList*. Combobox ini nantinya akan digunakan untuk memilih warna form yang akan dijadikan sebagai pilihan warna background form.



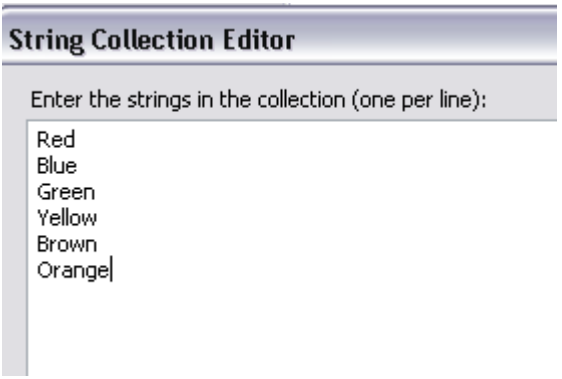


Komponen Combobox digunakan untuk menampilkan deretan pilihan item, dan hanya dapat dipilih satu item oleh pengguna. Sedangkan property *DropDownStyle* berfungsi agar combobox tidak dapat diisi item baru oleh pengguna.

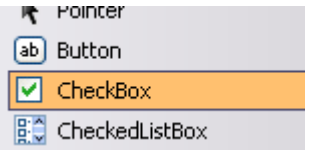
Selanjutnya klik button kecil di dalam property *Items* untuk menambahkan item di dalam combobox tersebut.

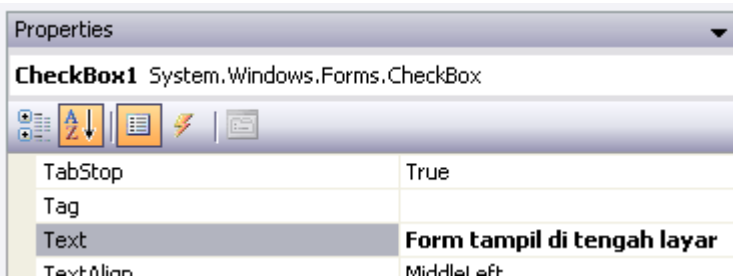


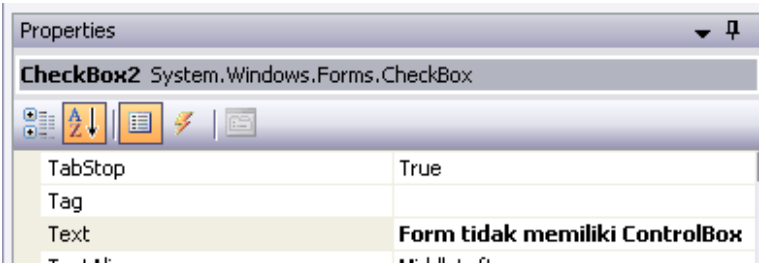
Lalu isikan item berikut ke dalam item combobox.



c. Dua buah komponen *Checkbox* tepat di bawah komponen label dan combobox. Set property *Text* di checkbox pertama dengan nilai *Form tampil di tengah layar*. Sedangkan checkbox kedua property *Text* diisi dengan nilai *Form tidak memiliki Control Box.*

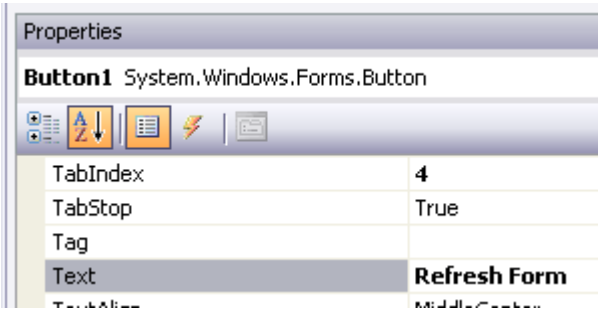


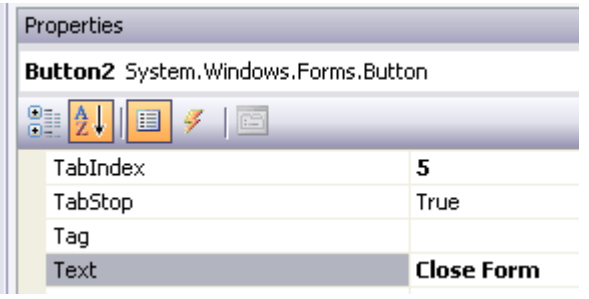




Hampir seluruh komponen yang memiliki sifat *display* di form memiliki property *Text*. Property tersebut memiliki fungsi yang sama yaitu sebagai display keterangan dari komponen yang bersangkutan.

d. Berikutnya, tempatkan dua buah komponen button, set property *Text* di button pertama dengan nilai *Refresh Form* dan di button kedua dengan nilai *Close Form*.





3. Kemudian, dobel klik pada pada form dan di dalam prosedur *Form_Load* ketikkan listing berikut ini :

```
ComboBox1.SelectedIndex = 0
```

Prosedur *Form_Load* akan dieksekusi pada saat form pertama kali dijalankan. Listing yang diketikkan tersebut berfungsi untuk mengarahkan item combobox langsung ke item pertama, sehingga combobox tidak terlihat kosong pada saat awal program dijalankan.

4. Setelah selesai, kembalikan mode display ke mode Form dengan melakukan dobel klik Form1 di Solution Explorer (atau dengan menekan tombol Shift+F7). Lalu dobel klik di button *Close Form*, dan di dalam prosedur *Button2_Click* ketikkan listing berikut ini yang berfungsi untuk menutup form :

```
Close()
```

5. Kemudian kembali lagi ke mode form, dan dobel klik di button *Refresh Form* untuk mengetikkan listing terakhir berikut ini di dalam prosedur *Button1_Click* :

```
Me.BackColor = _
    Color.FromName(ComboBox1.Text)
If CheckBox1.Checked Then
    Me.CenterToScreen()
Else
    Me.StartPosition = _

FormStartPosition.WindowsDefaultLocation
End If
Me.ControlBox = Not CheckBox2.Checked
```

Keterangan Listing :

Kata kunci *Me* merupakan pengganti dari form yang sedang aktif (pada contoh yang lain, Me bisa berarti komponen yang sedang aktif), sehingga pada konteks contoh ini, penggunaan Me

mengacu pada penggunaan form. Di dalam listing tersebut, form diatur propertynya sebanyak tiga kali. Yang pertama adalah mengubah warna latar belakang form (*BackColor*) dengan warna yang berasal dari combobox. Yang kedua mengatur posisi di tengah layar (*CenterToScreen*), dengan mengambil nilai checkbox yang pertama. Jika checkbox tersebut diklik, maka form akan diarahkan tepat di tengah layar. Sedangkan pengaturan yang terakhir adalah mengatur property Control Box (tiga icon default (minimize, maximize dan close) yang terletak di pojok kanan atas tiap form ), apakah akan ditampilkan atau tidak. Karena property tersebut memiliki tipe *boolean* (untuk mengetahui keterangan mengenai tipe data secara lengkap bisa dilihat di lampiran appendix), maka nilainya bisa langsung diambil dari kondisi (state) dari checkbox (dengan menggunakan operator *not* yang berarti negasi dari kondisi yang ada).

6. Untuk melakukan pengujian pada aplikasi tersebut, jalankan program dengan menekan tombol F5. Hasil dari aplikasi tersebut adalah sebagai berikut :

7. Untuk menguji lebih lanjut, pilihlah warna yang berbeda dari combobox, dan cobalah untuk melakukan penggantian warna dan melakukan kombinasi klik pada kedua checkbox sehingga pada saat ditekan button *Refresh Form* dapat terlihat efek dari pengaturan tiap komponen tersebut.

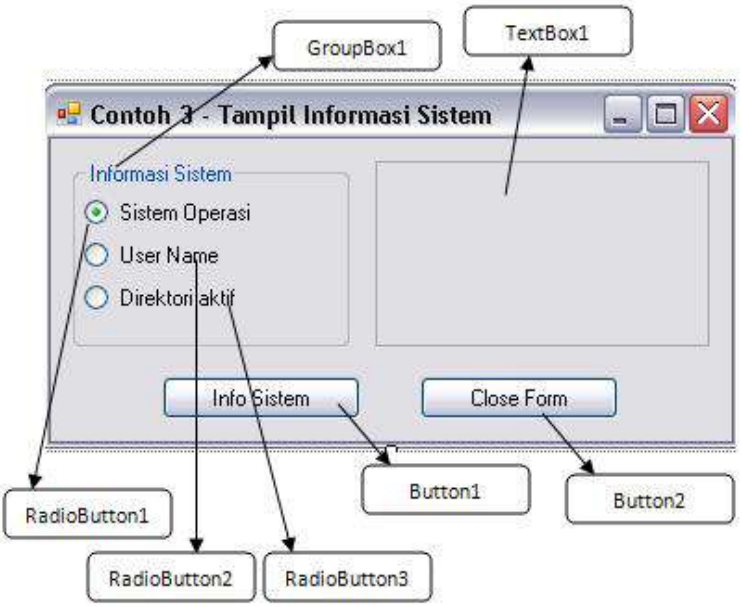
Saran Pengembangan :
Cobalah untuk mengisikan combobox yang berisi warna dengan berbagai jenis warna yang ada di dalam opsi Visual Basic. Pengisian tersebut juga dapat dilakukan di mode *run time* sehingga isi dari combobox akan sangat bervariasi.

Contoh 3 : Menampilkan Informasi Sistem

Level : Pemula
Tujuan :

1. Membuat form yang dapat diatur tampilannya.
2. Mengenalkan komponen Radiobutton dan GroupBox serta penggunaan logika perulangan *For Each* untuk melakukan pengecekan dalam sebuah *container*
3. Mengenalkan penggunaan kata kunci *My* untuk mendapatkan informasi sistem
4. Mengenalkan logika percabangan *If.. then..else..end if*.
5. Mengenalkan logika percabangan *Select Case…end select*

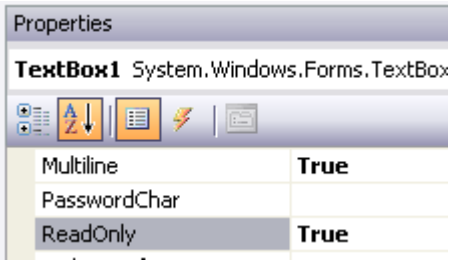
Desain form awal :



Langkah penyelesaian :

1. Seperti halnya contoh sebelumnya, buat sebuah solution baru dengan nama *Contoh3*.
2. Pada form yang tersedia, drag komponen-komponen berikut ini :

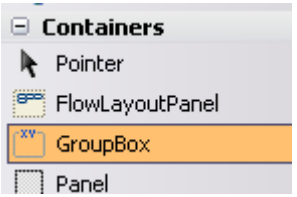
a. Sebuah textbox, lalu set property *ReadOnly* menjadi *True* seperti pada gambar berikut :

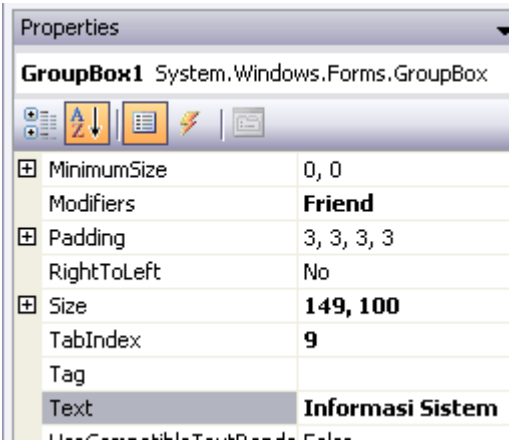


Property *Read Only* berfungsi agar komponen textbox hanya berfungsi sebagai penampil keterangan dan tidak dapat diinputkan nilai oleh pengguna.

b. Dua buah komponen button, dan untuk button pertama, set property *Text* di button pertama menjadi *Info Sistem* dan di button yang kedua menjadi *Close Form*.

c. Berikutnya drag sebuah komponen *GroupBox* yang terletak di tab *Container* di dalam Toolbox dan set property *Text* menjadi *Informasi Sistem*.



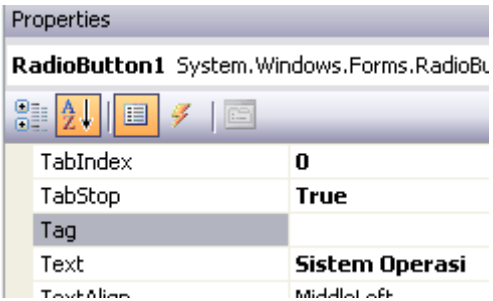


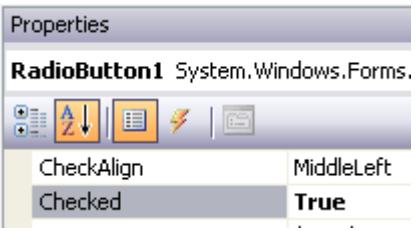
Komponen groupbox digunakan untuk mengelompokkan komponen lain agar menjadi sebuah grup tertentu. Dalam contoh ini, groupbox digunakan untuk melakukan pengelompokan pada komponen *RadioButton* agar dalam metode pengaksesannya dapat lebih mudah dilakukan.

d. Lalu tempatkan tiga buah komponen *RadioButton* di dalam groupbox yang sudah tersedia. Untuk tiap radiobutton

yang sudah ditempatkan, set property *Text* masing-masing sebagai berikut :

i. Radiobutton1 **à** Sistem Operasi dan property *Checked* diset menjadi *True*.





ii. Radiobutton2 **à** User Name

iii. Radiobutton3 **à** Direktori Aktif

Komponen radiobutton memiliki sifat yang mirip dengan combobox, yaitu digunakan untuk memilih satu item dari sebuah kelompok. Umumnya digunakan untuk sebuah kelompok pilihan yang memiliki sifat konstan atau tetap.

3. Langkah berikutnya, dobel klik pada button *Close Form* dan ketikkan listing berikut di prosedur *Button2_Click* untuk menutup form pada saat aplikasi berjalan :

```
Close()
```

4. Setelah itu, kembali lagi di mode form (dengan menekan shortcut Shift+F7), lalu dobel klik di button *Info Sistem*, lalu ketikkan listing berikut di dalam prosedur *Button1_Click* :

```
TextBox1.Clear()
Dim xPesan As String
For Each i As RadioButton In _
    GroupBox1.Controls
    If i.Checked Then
        Select Case i.Text
            Case "Sistem Operasi"
                xPesan = _
                    My.Computer.Info.OSFullName
            Case "User Name"
                xPesan = My.User.Name
            Case Else
                xPesan = _
                    My.Computer.FileSystem. _
```

```
                    CurrentDirectory
        End Select
        TextBox1.Text = i.Text & " : " & _
            vbCrLf & xPesan
    End If
Next
```

Keterangan Listing :

Di dalam listing tersebut, yang pertama kali dilakukan adalah membersihkan isi textbox dengan perintah *TextBox1.Clear.* Berikutnya, didefinsikan sebuah variabel bantu dengan nama *xPesan* yang memiliki tipe data *string* (berisikan teks).

Selanjutnya dilakukan perulangan dengan menggunakan perintah *For Each…* yang berfungsi untuk melakukan iterasi atau perulangan terhadap seluruh komponen yang terdapat di dalam groupbox (dalam contoh ini komponen yang terdapat dalam groupbox hanya bertipe radiobutton). Kemudian di dalam perulangan tersebut dilakukan pengecekan radiobutton yang sedang dalam keadaan terpilih (*i.checked*). Pengecekan dilakukan dengan menggunakan logika percabangan *if…then…else…end if* (sintaks percabangan ini dapat dilihat di appendix).

Dari radiobutton yang terpilh, kemudian dicek lagi dengan menggunakan percabangan *Select Case* untuk mendapatkan radiobutton mana yang sedang dipilih. Penggunaan *Select Case* digunakan karena kondisi dilakukan dari sebuah pemilihan tipe yang sama tetapi memiliki pilihan lebih dari dua (teks dari radiobutton).

Setelah terdeteksi radiobutton yang sedang dipilih, berikutnya ditampilkan informasi sistem dengan menggunakan kata kunci *My*. Kata kunci ini jika digabungkan dengan property *Computer* dapat menampilkan informasi sistem dari komputer, sedangkan jika digabungkan dengan property *User* dapat menampilkan

informasi user yang sedang login saat itu. Kata kunci *My* nantinya juga bisa digunakan untuk kepentingan lain (di contoh yang berbeda dapat digunakan untuk melakukan pengecekan keberadaan file dan penyimpanan file).
Kata kunci *vbCrLf* (Visual Basic Carriage Return Line Feed) berfungsi untuk menyisipkan penggantian baris di sebuah teks. Sedangkan tanda ampersand (&) digunakan sebagai penghubung antar variabel bertipe string dalam sebuah teks.

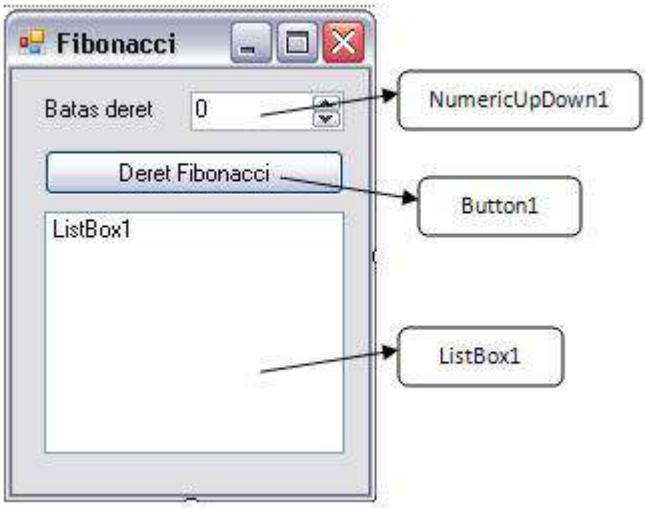
5. Untuk melakukan pengujian terhadap contoh yang sudah dikerjakan, klik icon *Start Debugging* ( ) atau tekan tombol F5. Hasil akhir yang didapat adalah sebagai berikut :

Saran Pengembangan :
Cobalah untuk menampilkan lebih banyak lagi informasi dari sistem, seperti besarnya memori jenis prosesor yang digunakan dan lainnya. Untuk menampilkan informasi tersebut dapat digunakan class WMI (Windows Management Instrumentation)

Contoh 4 : Deret Fibonacci

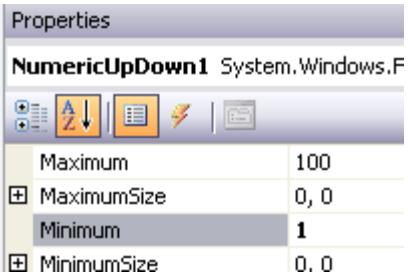
Level : Pemula
Tujuan :
1. Membuat deret bilangan Fibonacci
2. Menggunakan logika perulangan lebih lanjut
3. Mengenal penggunaan komponen Listbox dan NumericUpDown
4. Mengenal penggunaan array dalam logika pemrograman

Desain form awal :



Langkah penyelesaian :

1. Buat sebuah solution baru dengan nama *Contoh4*
2. Di dalam form yang tersedia, drag komponen-komponen berikut :
    a. Sebuah label dengan property *Text* berisi *Batas Deret*.
    b. Sebuah komponen *NumericUpDown* yang memiliki property *Minimal* bernilai 1.



Komponen numericupdown digunakan untuk menampung entrian bertipe numerik dari pengguna. Property penting dari komponen ini adalah property *Minimum* yang berfungsi sebagai pembatas nilai minimal yang dapat dientrikan dan property *Maximum* yang berfungsi sebagai pembatas nilai maksimal.

c. Sebuah button dengan property *Text* berisi *Deret Fibonacci*.
d. Sebuah komponen *ListBox* di bawah komponen button yang sudah didrag sebelumnya.

Komponen listbox memiliki sifat yang hampir sama dengan combobox, yaitu memilih item dari sebuah kumpulan item. Perbedaannya adalah di dalam komponen listbox dapat dilakukan pemilihan item lebih dari satu jika property *SelectionMode* diset menjadi jenis *Multi*.

3. Lalu dobel klik di button yang ada, dan ketikkan listing berikut di dalam prosedur *Button1_Click* :

```
Dim w, y, z As Integer
y = 0
z = 1
ListBox1.Items.Clear()
ListBox1.Items.Add(y)
ListBox1.Items.Add(z)
For x As Integer = _
    1 To NumericUpDown1.Value
    w = y + z
    y = z
    z = w
    If w > NumericUpDown1.Value _
        Then Exit For
    ListBox1.Items.Add(w)
Next
```

Keterangan Listing :
Deret Fibonacci merupakan deret bilangan yang memiliki rumus bahwa bilangan berikutnya adalah hasil penjumlahan dari dua bilangan sebelumnya, kecuali dua bilangan pertama yang selalu diisi dengan nilai nol dan satu. Sebagai contoh : 0,1,1,2,3,5 dan seterusnya.

Deret bilangan ini menjadi semacam "bahan latih dasar" bagi tiap programmer pemula untuk mengimplementasikan logika pemrograman dasar.
Dalam listing di contoh ini, dua bilangan awal diset pada awal untuk dimasukkan ke dalam listbox (dengan menggunakan perintah *Items.Add*). Kemudian diimplementasikan logika perulangan sebanyak batas deret bilangan yang telah dientrikan di dalam komponen numericupdown dengan menggunakan tiga variabel bantu yaitu w, y dan z.
Variabel w berfungsi untuk menampung hasil penjumlahan dua bilangan sebelumnya dari deret yang tersedia yaitu variabel y dan z.
Dan pada saat hasil penjumlahan tersebut telah melebihi batas yang telah ditentukan (ditandai dengan pengecekan menggunakan percabangan *If*), maka perulangan dikeluarkan dari proses dengan menggunakan perintah *Exit For*.
Sebelum perulangan dilakukan, terlebih dulu komponen listbox dibersihkan isinya dengna menggunakan perintah *Items.Clear.* Hal ini dilakukan agar saat nilai batas deret diganti dan diklik ulang button yang tersedia, maka listbox hanya akan berisi nilai yang dimaksudkan.

4. Untuk menguji hasil aplikasi, dapat ditekan tombol F5. Dan sebagai tester dapat dientrikan beragam angka (dengan nilai maksimal 100) di dalam numericupdown sebelum melakukan klik pada button.


Fibonacci
Batas deret
10
Deret Fibonacci
0
1
1
2
3
5
8

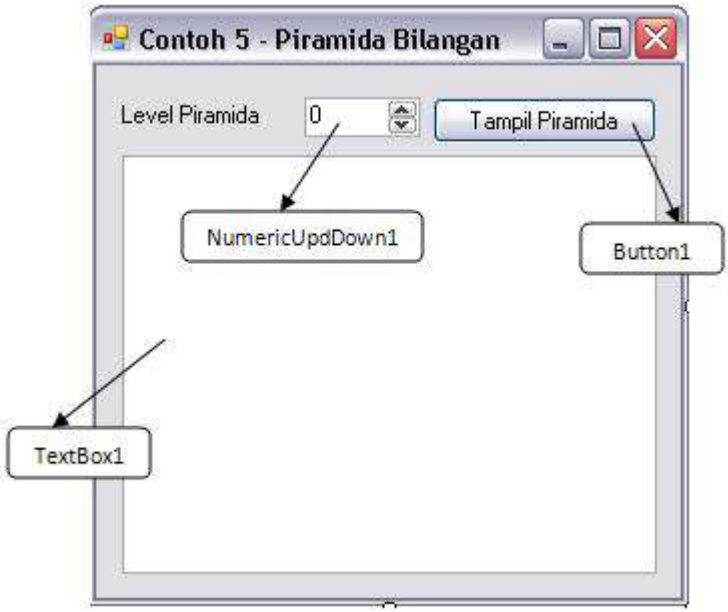

Fibonacci
Batas deret
24
Deret Fibonacci
0
1
1
2
3
5
8
13
21

Saran Pengembangan :
Dalam aplikasi tersebut juga dapat ditambahkan satu parameter pilihan untuk berapa banyak jumlah bilangan fibonacci yang akan ditampilkan dalam aplikasi.

Contoh 5 : Piramida bilangan

Level : Pemula
Tujuan :
1. Membuat model piramida bilangan dengan menggunakan Textbox sebagai komponen peraga
2. Mengenalkan penggunaan Textbox Multiline.
3. Mengenalkan manipulasi string sederhana dalam Visual Basic
4. Menggunakan perulangan *Do While*

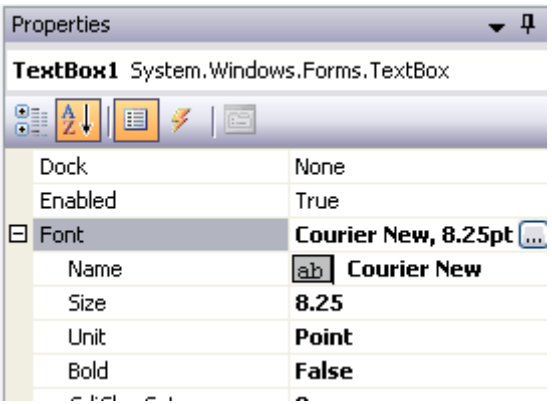
Desain form awal :



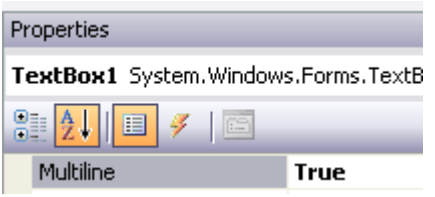
Langkah penyelesaian :

1. Buat sebuah solution baru dengan nama *Contoh5*
2. Di dalam form tersebut, atur ukuran sedemikian rupa sehingga menyerupai desain form awal, kemudian drag komponen berikut ini :
    a. Sebuah label dengan property *Text* berisi *Level Piramida*

b. Sebuah numericupdown dengan property *Minimal* bernilai 2
c. Sebuah button dengan property *Text* berisi *Tampil Piramida*
d. Sebuah textbox dengan property *Multiline* bernilai *True*. Di dalam textbox tersebut, set juga property *Font* menjadi *Courier New* dengan style *Bold*.



Pemilihan font Courier New agar pada saat piramida ditampilkan huruf dapat terlihat rata. Hal ini disebabkan font Courier New memiliki sifat unik yaitu memiliki lebar yang sama untuk tiap karakter.



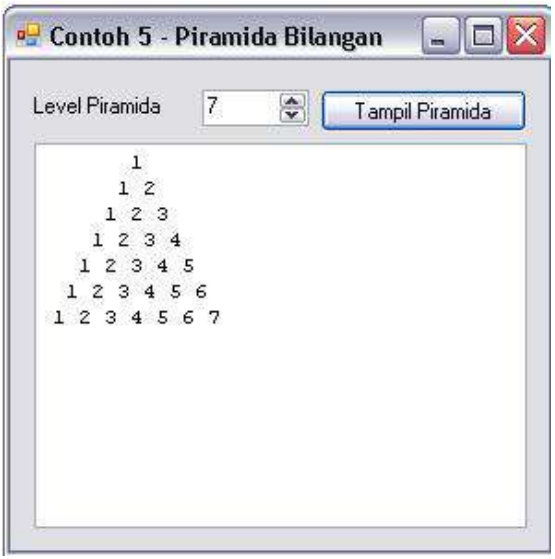
Property *Multiline* di dalam textbox akan mengaplikasi textbox sebagai sebuah isian (atau juga tampilan) yang mampu menampung lebih dari satu baris dalam satu waktu.

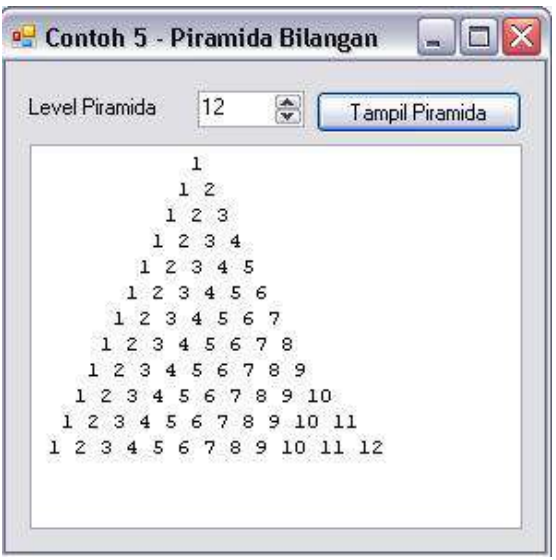
3. Kemudian dobel klik pada button *Tampil Piramida* dan ketikkan listing berikut ini :

```
Dim x, y, z As Integer
x = 1 : y = NumericUpDown1.Value
TextBox1.Clear()
Do While x <= y
    z = 1
    TextBox1.Text = _
    TextBox1.Text & Space(y - x)
    Do While z <= x
        TextBox1.Text = _
        TextBox1.Text & " " & z
        z += 1
    Loop
    TextBox1.Text = _
    TextBox1.Text & vbCrLf
    x += 1
Loop
```

Keterangan Listing :
Membuat piramida bilangan (seperti halnya deret Fibonacci) merupakan bahan latihan dasar dalam logika pemrograman. Pembuatan piramida bilangan akan menguji penggunaan *nested looping* (perulangan bersarang atau perulangan di dalam perulangan yang lain). Hasil akhir dari piramida bilangan adalah deret bilangan yang tiap barisnya akan bertambah sesuai urutan nomor baris dan membentuk sebuah segitiga sama kaki layaknya piramida dua dimensi.
Dalam contoh ini, dilakukan perulangan dua kali yaitu yang pertama untuk menempatkan deretan angka dalam satu baris, dan yang kedua (nested) akan menempatkan bilangan sejumlah nomor urut baris. Dalam tiap deret angka, dipisah sebuah spasi agar terlihat rapi. Sedangkan untuk melakukan perhitungan spasi agar deret pertama dapat berada di tengah (sebagai puncak piramida), maka di perulangan pertama dilakukan perhitungan spasi dengan menggunakan fungsi *Space* yang berguna untuk menempatkan spasi sebanyak *counter*.
Perlu diperhatikan bahwa perulangan dengan *do While* membutuhkan counter sebagai pembatas kapan perulangan akan berakhir. Dalam contoh ini, counter yang digunakan adalah variabel x. Sedangkan variabel y merupakan variabel yang berfungsi sebagai batas atas dari total perulangan yang harus dilakukan. Batas atas tersebut diambil dari inputan pengguna di dalam numericupdown.

4. Untuk melakukan pengujian dari aplikasi ini, tekan F5 untuk mengeksekusi program. Hasil akhir yang didapat dapat dilihat pada gambar berikut :





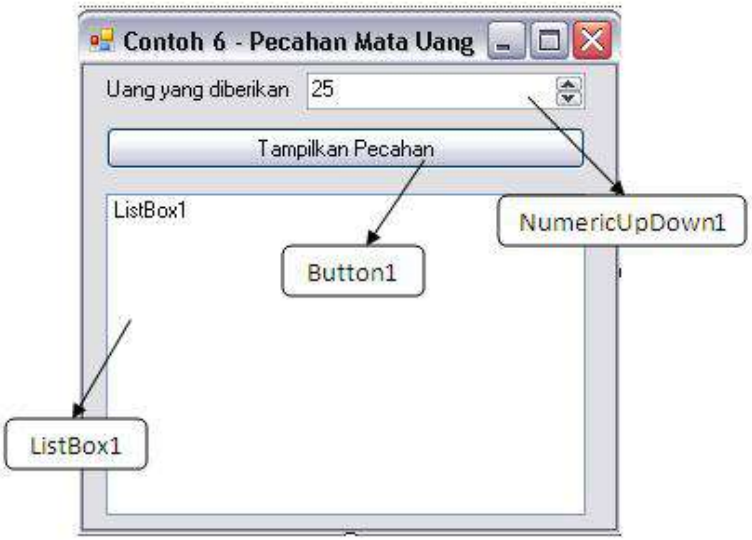
Saran Pengembangan :

Dapat ditambahkan potongan listing yang nantinya dapat menyebabkan piramida bilangan benar-benar berada di tengah textbox untuk puncak bilangannya. Sehingga nantinya akan terbentuk sebuah piramid tepat di tengah textbox.

Contoh 6 : Pecahan mata uang

Level : Pemula
Tujuan :
1. Membuat aplikasi untuk memecah besaran mata uang ke pecahan yang telah tersedia.
2. Menggunakan perulangan lebih lanjut
3. Mengenal penggunaan array

Desain form awal



Langkah penyelesaian :

1. Buat solution baru dengan nama *Contoh6*
2. Di dalam form yang ada, drag komponen berikut :
    a. Sebuah label dengan property *Text* berisi *Uang yang diberikan*
    b. Sebuah numeriupdown dengan property *Minimum* bernilai 25 dan property *Maximum* bernilai 5000000 (lima juta).
    c. Sebuah listbox

d. Sebuah button dengan property *Text* bernilai *Tampilkan Pecahan*

3. Berikutnya, dobel klik di button *Tampilkan Pecahan* dan ketikkan listing berikut ini :

```
Dim xpecahan() As Integer = _
    {100000, 50000, 25000, 10000, _
    5000, 1000, 500, _
    200, 100, 50, 25}
Dim xhasil, xtemp, i As Integer
xhasil = NumericUpDown1.Value
ListBox1.Items.Clear()
For i = 0 To xpecahan.GetUpperBound(0)
    xtemp = Int(xhasil / xpecahan(i))
    xhasil -= xpecahan(i) * xtemp
    ListBox1.Items.Add _
        ("Pecahan : " & _
        xpecahan(i) & " sebanyak : " _
        & xtemp)
Next
```

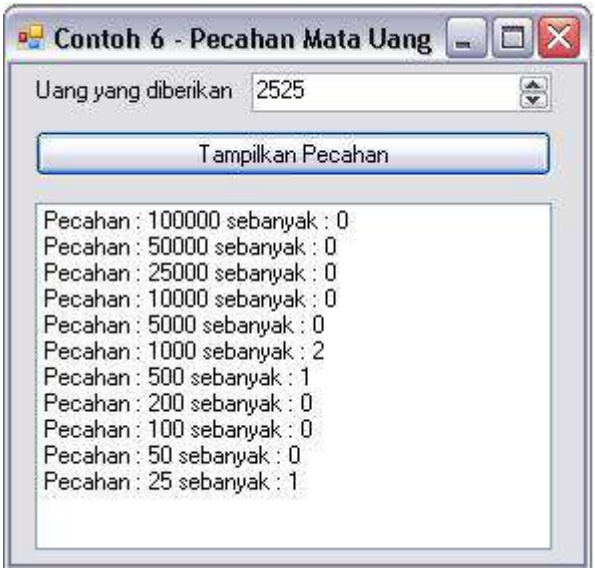
Keterangan Listing :

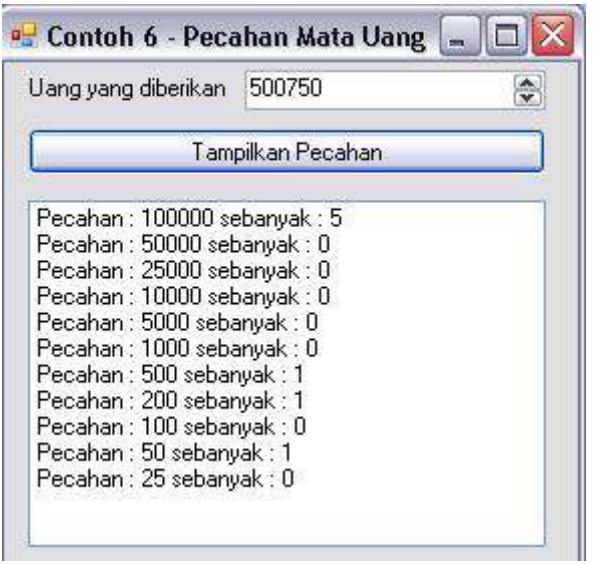
Pada listing untuk menampilkan pecahan mata uang (rupiah) didemonstrasikan penggunaan variabel jenis array di dalam sebuah aplikasi (untuk lebih lanjut memahami array dapat dilihat di appendix mengenai variabel). Dalam listing tersebut terdapat tiga buah variabel yaitu : variabel *xpecahan* berjenis array yang berguna untuk menampung jenis pecahan mata uang (dari 25 hingga 100.000), variabel *xtemp* yang berguna sebagai penampung sementara dalam proses pembagian dari jumlah uang menjadi pecahan dan yang terakhir adalah variabel *xhasil* sebagai penampung jumlah mata uang yang tersisa.

Di proses awal variabel *xhasil* diisi dengan jumlah uang yang diinputkan di dalam numericupdown, kemudian dibagi sesuai

dengan jenis pecahan yang ada (dengan menggunakan proses looping). Lalu jika terdapat sisa dari hasil pembagian tersebut, maka akan dibagi lagi ke pecahan yang lebih kecil dan seterusnya hingga selesai.

4. Lalu eksekusi program dengan menekan tombol F5 dan hasil yang didapat adalah sebagai berikut :



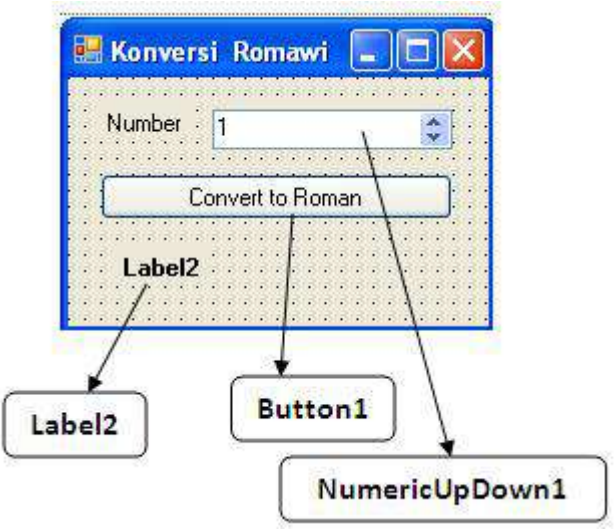


Saran Pengembangan :
Dapat ditambahkan pecahan baru Rp. 2000 di dalam aplikasi perhitungan yang ada.

Contoh 7 : Konversi Angka Romawi

Level : Pemula
Tujuan :
1. Membuat aplikasi untuk melakukan konversi angka arab ke angka romawi.
2. Menggunakan percabangan lebih lanjut
3. Mengenal konversi tipe angka bulat dan pembulatannya
4. Mengenal pembuatan dan penggunaan function

Desain form awal :



Langkah penyelesaian :

1. Buat sebuah Solution baru dengan nama *Contoh7*.
2. Di dalam form yang tersedia, drag komponen-komponen berikut ini :
    a. Dua buah label dengan rincian :
        i. Untuk label yang pertama set property *Text* dengan isi *Number*
        ii. Label yang kedua set property *Font* **à** *Bold* menjadi *true*.

b. Sebuah button dengan isi property *Text* yang diisi nilai *Convert to Roman*
c. Sebuah numeriupdown dengan property *Minimum* bernilai 1 dan property *Maximum* bernilai 100000 (seratus ribu).

3. Kemudian dobel klik di bagian form yang kosong, sehingga menuju ke prosedur *Form_Load* lalu ketikkan listing berikut :

```
Label2.Text = "Roman number : "
```

4. Selanjutnya masih di dalam bagian pengetikan listing, ketikkan fungsi untuk membantu penyelesaian masalah konversi angka :

```
Function xResult(ByVal x As Integer, _
      ByVal y As Integer, _
      ByVal st As String) As String
      Return StrDup(CInt _
              (Int((x / y))), st)
End Function
```

Keterangan Listing :
Dalam fungsi dengan nama *xResult* ini akan dikembalikan sebuah nilai balik yang nantinya berisi rangkaian karakter (diwakili dengan variabel dengan nama *st*) sebanyak hitungan tertentu (diwakili dengan perhitungan *CInt(Int(x/y)*. Fungsi *CInt* melakukan konversi sebuah angka ke dalam bilangan bulat tipe integer, sedangkan fungsi *Int* juga melakukan hal yang sama tetapi ke tipe *Short*. Dan karena fungsi *StrDup* yang berfungsi

menduplikasi karakter tersebut hanya menerima inputan tipe integer, maka konversi dilakukan dua kali.

5. Kemudian ketikkan fungsi yang kedua yakni fungsi *ToRoman* yang akan menjadi otak dari program ini :

```
Function ToRoman(ByVal X As Integer) As String
Dim yResult As String
'bilangan ribuan
yResult = xResult(X, 1000, "M")
X = X - (CInt(Int(X / 1000) * 1000))
'ribuan dan ratusan
If X >= 900 Then
    yResult = yResult & "CM"
ElseIf X >= 500 And X < 900 Then
    yResult = yResult & "D" & _
    xResult(X - 500, 100, "C")
ElseIf X >= 400 And X < 500 Then
    yResult = yResult & "CD"
Else
    yResult = yResult & _
    xResult(X, 100, "C")
End If
X = X - (CInt(Int(X / 100) * 100))
'ratusan dan puluhan
If X >= 90 Then
    yResult = yResult & "XC"
ElseIf X >= 50 And X < 90 Then
    yResult = yResult & "L" & _
    xResult(X - 50, 10, "X")
ElseIf X >= 40 And X < 50 Then
    yResult = yResult & "XL"
Else
    yResult = yResult & _
    xResult(X, 10, "X")
End If
```

```
X = X - (CInt(Int(X / 10) * 10))
'satuan
If X >= 9 Then
    yResult = yResult & "IX"
ElseIf X >= 5 And X < 9 Then
    yResult = yResult & "V" & _
    xResult(X - 5, 1, "I")
ElseIf X >= 4 And X < 5 Then
    yResult = yResult & "IV"
Else
    yResult = yResult & _
    xResult(X, 1, "I")
End If
ToRoman = yResult
End Function
```

Keterangan Listing :

Fungsi *ToRoman* merupakan latihan yang sangat bagus untuk memperdalam penggunaan percabangan *if* dikarenakan dalam konversi ke angka romawi terdapat banyak pengecualian. Di tiap bagian percabangan baik ribuan, ratusan hingga ke satuan, konversi angka romawi memiliki pengecualian di bilangan sebelum bilangan pembatas. Sebagai contoh dalam pembatas bilangan satuan yakni sepuluh, maka di angka sembilan akan memiliki pengecualian dalam penulisannya, yakni dikurangi satu. Begitu pula dengan pembatas bilangan ratusan dan ribuan. Dari tiap kondisi yang normal, maka konversi yang dilakukan sangat sederhana karena hanya menduplikasi angka romawi sebanyak bilangan pembaginya. Proses duplikasi ini yang dibantu dengan cara memanggil fungsi yang sebelumnya yakni *xResult*.

6. Yang terakhir adalah dengan melakukan dobel klik pada satu-satunya button dalam form lalu mengisikan listing berikut ini :

```
Label2.Text = "Roman number : " & _
    ToRoman(NumericUpDown1.Value)
```

Keterangan Listing :
Di dalam button hanya akan memanggil fungsi *ToRoman* dan menampilkan ke dalam label yang tersedia.

7. Hasil dari eksekusi program tampak seperti pada gambar berikut ini :

Saran Pengembangan :
Untuk latihan, program ini dapat dikembangkan dengan membuat program yang memungkinkan untuk konversi dari

angka romawi ke dalam angka arabik. Sehingga dapat dikembangkan sebuah program yang bertipe *parsing*.

Pemrograman Visual Lanjutan

Contoh 1 : Random Star Screensaver

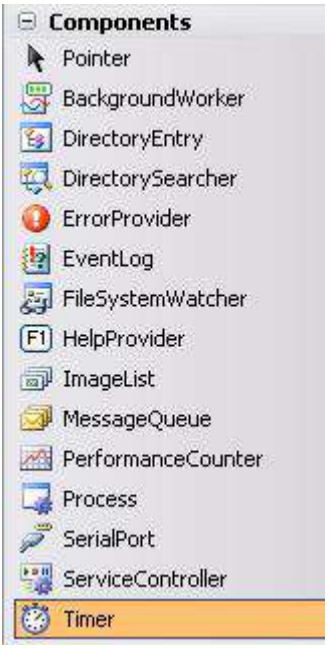
Level : Menengah
Tujuan :

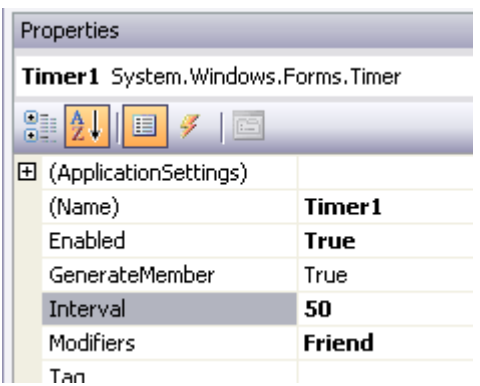
1. Membuat aplikasi screensaver berupa karakter bintang (*) yang secara acak (baik posisi maupun warna) muncul di seluruh layar.
2. Mengenal penggunaan bilangan acak
3. Mengenal setting posisi dan warna dari sebuah komponen secara *runtime*
4. Mengenal penggunaan *Timer*
5. Mengenal pendeteksian kursor mouse
6. Menggunakan aplikasi sebagai screensaver dalam Windows

Langkah penyelesaian :

1. Desain form awal (dengan nama *StarScreenSaver*) berupa sebuah form dengan sebuah label (posisi terserah) dengan property *Text* berisi karakter bintang **à** ***** dengan ukuran font sebesar 16 (bisa juga diset dengan ukuran yang lebih besar).
2. Sedangkan untuk form yang tersedia, set property *FormBorderStyle* menjadi *None* dan property *BackColor* menjadi *Black* (warna hitam). Set juga property *WindowState* menjadi *Maximized*.

3. Kemudian drag sebuah komponen *Timer* (terletak di tab *Component* dalam Toolbox) dan set property *Enabled* menjadi *True* dan property *Interval* bernilai 50.





Komponen timer merupakan komponen yang cara kerjanya mirip dengan *stopwatch* dengan melakukan

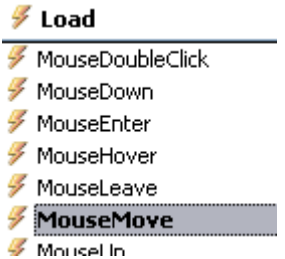
perhitungan waktu dan mengerjakan apapun yang ada di dalam prosedur *Tick* sesuai dengan interval yang telah diberikan. Nilai interval memiliki satuan milisecond atau milidetik (1000 milidetik sama dengan 1 detik).

4. Selanjutnya dobel klik pada form, untuk menuju ke prosedur *Form_Load*, lalu ketikkan listing program berikut ini :

```
Me.Cursor.Hide()
```

Keterangan Listing :
Perintah *Cursor.Hide* berfungsi untuk menyembunyikan kursor mouse, sehingga pada saat screensaver bekerja, kursor tidak akan terlihat oleh pengguna.

5. Kemudian di pilihan combobox *event*, pindahkan ke prosedur *MouseMove* (untuk mendeteksi pergerakan mouse pada form saat dijalankan) dan ketikkan listing berikut di dalamnya :




```
If Not aktif Then
    posisi = New Point(e.X, e.Y)
    aktif = True
Else
    If Math.Abs(e.X - posisi.X) > 15 Or _
        Math.Abs(e.Y - posisi.Y) > 15 Then
        Close()
    End If
End If
```

Keterangan Listing :

Seperti halnya screensaver lain, maka form ini akan keluar dengan sendirinya saat mouse digerakkan. Tetapi pada saat awal dijalankan, pergerakan mouse di Windows melakukan sedikit "tipuan", sehingga saat mouse diam pun dianggap melakukan gerakan dengan posisi 0,0. Karenanya, tipuan tersebut diatasi dengan menggunakan "penangkapan" posisi awal mouse yang kemudian dilakukan pengecekan, bahwa mouse minimal bergerak sejauh 15 pixel untuk dianggap bahwa mouse telah digerakkan oleh pengguna.

Penangkapan tersebut membutuhkan dua variabel bantu yang dideklarasikan di awal form yaitu variabel *aktif* (untuk menandakan bahwa form telah berjalan) dan variabel *posisi* yang berfungsi untuk menangkap posisi akhir mouse.

6. Lalu tepat di bawah deklarasi *Public Class Form …* ketikkan deklarasi dua variabel bantu berikut :

```
Dim aktif As Boolean = False
Dim posisi As Point
```

Keterangan Listing :
Variabel *aktif* yang bertipe boolean merupakan tanda awal bahwa form telah dijalankan. Isi dari variabel diisi dengan nilai default *False* dan nantinya akan langsung diisi nilai kebalikkannya (yaitu *True*) pada saat mouse mulai bergerak (lihat di poin sebelumnya). Sedangkan variabel *posisi* merupakan variabel yang bertipe *point* yang berfungsi untuk menampung koordinat awal dari mouse saat form dijalankan.

7. Selanjutnya, kembalilah ke mode form (dengan menekan tombol Shift+F7) dan dobel klik di timer, dan ketikkan listing berikut :

```
Dim acak As New Random
With Label1
    .Top = acak.Next(Me.Height)
    .Left = acak.Next(Me.Width)
    .ForeColor = _
        Color.FromArgb(255, _
        acak.Next(255), acak.Next(255), _
        acak.Next(255))
End With
```

Keterangan Listing :
Variabel dengan nama *acak* merupakan variabel dengan tipe obyek *Random* yang nantinya dapat digunakan untuk melakukan

generate bilangan acak. Pada listing tersebut, bilangan acak yang pertama digunakan untuk mengacak koordinat x dari label dengan mengacak bilangan dengan batas atas tinggi dari form itu sendiri (acak.Next(Me.Height)). Perintah *Next* merupakan perintah untuk melakukan generate bilangan acak, dan parameter dalam kurung adalah batas atas bilangan acak yang boleh ada.

Dengan cara yang sama, maka dapat diacak pula koordinat y dari label dengan menggunakan batas atas lebar dari form. Dan karena property *WindowState* telah diset menjadi maximized, maka tinggi dan lebar form adalah resolusi dari layar itu sendiri.

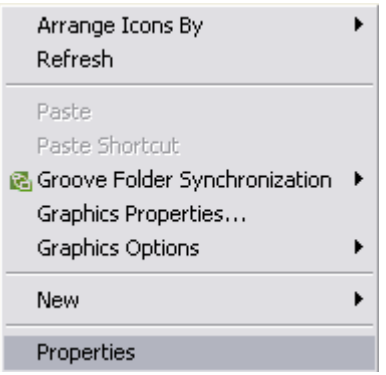
Bilangan acak berikutnya digunakan untuk melakukan pengacakan di warna label dengan menggunakan obyek *Color.FromArgb.* Seperti telah diketahui secara jamak, bahwa warna dasar di dalam ilmu komputer adalah RGB **à** Red, Green Blue. Sedangkan A dalam *Argb* adalah nilai alpha atau transparansi warna, yang diset menjadi 255 (absolut, sehingga warna terlihat jelas).

Kombinasi warna dasar RGB memiliki jangkauan nilai 0 – 255. Sebagai contoh, jika nilai RGB diset menjadi 0,0,0 maka akan terlihat warna hitam, dan jika diset menjadi 255,255,255 akan terlihat warna putih. Tetapi jika salah satu unsur diset menjadi absolut, maka akan menjadi warna dominan, misalkan : 255,0,0 maka akan menjadi warna merah, 0,255,0 akan menjadi warna hijau dan 0,0,255 akan menjadi warna biru. Sedangkan jika diset secara kombinasi misal : 50,35,90 maka akan terjadi warna campuran yang sulit untuk ditebak.

Dan dengan melakukan pengacakan seperti pada listing, maka label yang diacak tersebut akan muncul di koordinat yang tidak teratur, sekaligus akan memiliki warna yang acak pula. Efek dari pengacakan tersebut akan menjadi semacam efek bintang

berkedip dari latar belakang hitam (sebagai ganti dari nuansa malam).

8. Langkah berikutnya adalah mengeksekusi program tersebut. Jika program telah benar, maka akan terlihat layar yang gelap dengan kilauan karakter ***** yang muncul secara tidak beraturan didalamnya.
9. Kini setelah selesai, bukalah aplikasi Windows Explorer lalu carilah file *exe* yang telah terbentuk di dalam folder *bin*. Selanjutnya, ganti nama dari file exe tersebut menjadi file yang memiliki ekstensi *scr* (agar dikenali sebagai screensaver). Kemudian copykan ke dalam folder *Windows/System32*.
10. Untuk melakukan pengecekan, berpindahlah ke desktop Windows, dan klik kanan lalu pilih sub menu *Properties*.



11. Di dalam kotak dialog yang tersedia, pilih tab *Screensaver* dan carilah screensaver dengan nama solution yang telah didefinisikan sebelumnya. Untuk melakukan testing, klik tombol *Preview*. Jika semua berjalan dengan benar, maka Anda telah mendapatkan sebuah screensaver baru buatan sendiri !!

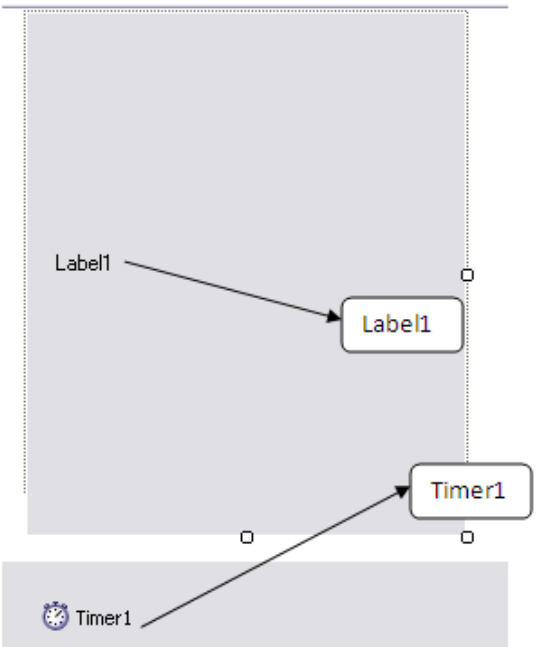
Saran Pengembangan :
Anda dapat mengganti karakter star (*) dengan sebuah gambar yang menarik sehingga screensaver yang terbentuk dapat terlihat lebih hidup.

Contoh 2 : Marquee Screensaver

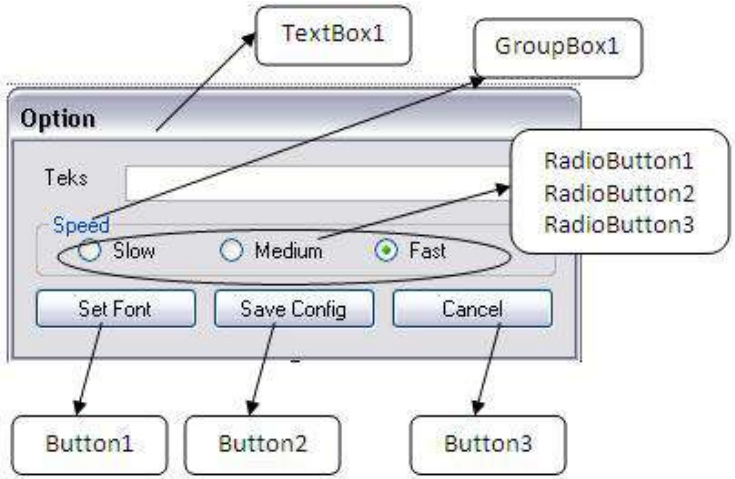
Level : Menengah
Tujuan :

1. Membuat aplikasi screensaver berupa rangkaian kata yang berjalan dari kiri ke kanan layar, kemudian kembali lagi (scrolling marquee)
2. Mengaplikasikan teknik penggunaan form majemuk
3. Mengaplikasikan teknik penyimpanan setting dalam bentuk file
4. Menggunakan komponen timer dalam aplikasi untuk animasi sederhana
5. Mengenal manipulasi array tipe string
6. Mengenal penggunaan prosedur dan function
7. Mengenal penggunaan FontDialog

Desain form pertama (*MarqueeScreensaver*) :



Desain form kedua :



Langkah penyelesaian awal :

1. Buat sebuah solution dengan dua buah form, yang pertama untuk screensaver utama dan yang kedua untuk melakukan pemilihan option di dalam screensaver tersebut.
2. Form yang pertama memiliki property sebagai berikut :
    a. FormBorderStyle : None
    b. WindowState : Maximized
    c. Name : MarqueeScreenSaver
    d. KeyPreview : True

Property ***KeyPreview*** digunakan untuk menangkap penekanan tombol di dalam form saat form dieksekusi. Jika bernilai *True*, maka penekanan tombol akan ditangkap, dan sebaliknya jika bernilai *False* maka penekanan tombol di dalam form akan diabaikan (kecuali untuk shortcut).

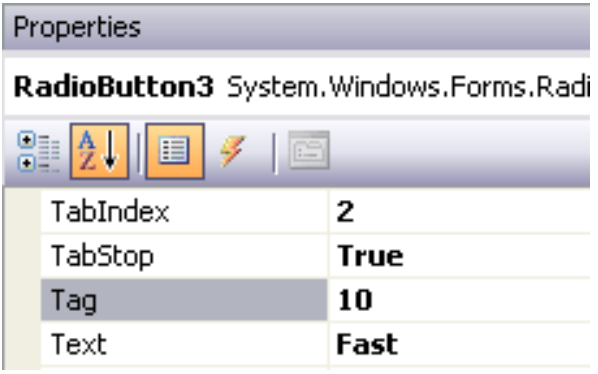
3. Sedangkan komponen yang terdapat di dalam form pertama yaitu :
    a. Label1
    b. Timer1 dengan setting property :
        i. Interval : 10
        ii. Enabled : True
4. Untuk form yang kedua, memiliki property :
    a. Name : MarqueeOption
    b. Text : Option
    c. ControlBox : False

Setting property ***ControlBox*** ditujukan agar form tersebut hanya memiliki satu "jalan keluar" yaitu dengan menekan button *Cancel*. Teknik ini sering digunakan agar pengguna tidak rancu pada saat form tersebut memiliki kondisi tertentu yang didalamnya memiliki proses "melempar nilai" sebuah variabel ke form lainnya.

5. Dan komponen yang terdapat di dalam form kedua adalah :
    a. Label1, dengan property *Text* berisi *Teks*
    b. Textbox1
    c. Groupbox1, dan didalamnya terdapat tiga komponen radiobutton yang masing-masing property *text* berisi *Slow, Medium* dan *Fast*. Tiap radiobutton tersebut memiliki setting property *Tag* sebagai berikut :
        i. Radiobutton *Slow* **à** 50
        ii. Radiobutton *Medium* **à** 25
        iii. Radiobutton *Fast* **à** 10

Property *Tag* sebenarnya merupakan property yang tidak memiliki fungsi khusus dan terdapat di hampir semua komponen yang memiliki sifat display di dalam sebuah form. Dalam contoh ini, property tersebut diisikan nilai yang nantinya akan dimanfaatkan agar pada saat proses pemilihan kondisi, tidak terlalu banyak percabangan yang harus dituliskan di dalam listing. Perlu diingat juga bahwa nilai property tag selalu berisi tipe data string, sehingga jika didalamnya akan terjadi perhitungan, disarankan untuk melakukan konversi

secara eksplisit ke dalam tipe data yang dimaksudkan, misal : Cint (untuk konversi ke tipe data integer), atau Cdate(untuk konversi ke tipe data tanggal).



d. Tiga buah button untuk *Set Font*, *Save Config* dan *Cancel*.

Langkah penyelesaian untuk form yang pertama (dengan asumsi telah diberi nama *MarqueeScreenSaver*) :

1. Dobel klik pada form yang ada, dan di dalam prosedur *MarqueeScreenSaver_Load* ketikkan listing berikut ini :

```
Me.Cursor.Hide()
With My.Computer.FileSystem
    If .FileExists(namaFile) Then
        RefreshScreen()
    Else
```

```
        Dim teks As String = _
       "speed=10" & vbCrLf & "font=verdana" & _
        vbCrLf & "fontsize=20" & vbCrLf & _
        "teks=Learning By Sample"
        .WriteAllText(namaFile, teks, True)
    End If
End With
```

Keterangan Listing :
Pada saat pertama kali screensaver dijalankan, maka akan dicek keberadaan file konfigurasi. Jika ternyata file konfigurasi sudah ditemukan, maka langkah berikutnya akan dijalankan prosedur *RefreshScreen* (akan dijelaskan di langkah selanjutnya) yang akan membaca file konfigurasi dan menerapkannya di dalam screensaver. Tetapi jika file konfigurasi belum ada, maka akan dibuat sebuah file konfigurasi baru dengan konfigurasi default.

2. Sedangkan untuk pendeklarasian variabel awal, ketikkan deklarasi variabel berikut tepat di bawah *Public Class ….*

```
Dim aktif As Boolean = False
Dim posisi As Point
Dim namaFile As String = "marquee.cfg"
```

Keterangan Listing :
Deklarasi variabel awal terdiri dari tiga variabel, yaitu variabel aktif yang bertipe boolean dan digunakan sebagai tanda saat screensaver pertama kali dijalankan dan variabel posisi (bertipe obyek *point*) yang berfungsi sebagai variabel bantu dalam melakukan pergerakan label.
Sedangkan variabel terakhir, yaitu variabel *namaFile* berfungsi untuk memberi nama default dari file konfigurasi yang akan

digunakan sebagai tempat penyimpanan setting dari screensaver tersebut.

3. Tepat di bawah prosedur tersebut, ketikkan prosedur baru bernama *RefreshScreen* dan sebuah function bernama *ambilkonfig* yang berisikan listing berikut :

```
Function ambilKonfig(ByVal indeks As Integer)
As String
Dim Konfigurasi() As String = _
    Split(My.Computer.FileSystem. _
          ReadAllText(namaFile), vbCrLf)
  Return Mid(Konfigurasi(indeks), _
  CStr(Konfigurasi(indeks)).IndexOf("=") + 2, _
           Konfigurasi(indeks).Length)
End Function

Public Sub RefreshScreen()
Timer1.Interval = ambilKonfig(0)
With Label1
    .Font = New Font(ambilKonfig(1), _
                     ambilKonfig(2))
    .Text = ambilKonfig(3)
End With
End Sub
```

Keterangan Listing :

Dari listing tersebut, terdapat sebuah function bernama *ambilkonfig* yang berfungsi untuk mengambil data dari file teks yang sudah didefinisikan, dan sebuah prosedur *RefreshScreen* yang digunakan untuk melakukan refresh terhadap screensaver setelah terjadi perubahan konfigurasi.

Di dalam function dapat dilihat bahwa yang pertama kali dilakukan adalah mendeklarasikan sebuah array string dengan

nama *Konfigurasi*. Array ini diambil dari file teks yang telah dideklarasikan sebelumnya (yaitu *marquee.cfg*) dengan menggunakan fungsi *Split*. Fungsi *Split* berfungsi untuk memecah sebuah string menjadi array string dengan menggunakan pemisah tertentu (dalam contoh ini pemisah yang digunakan adalah *vbCrLf* atau *Visual Basic Carriage Return Line Feed* atau lebih lazim disebut ganti baris). Dari tiap indeks array string nantinya akan diambil nilai yang berada setelah tanda "=" (ditandai dengan adanya fungsi *Mid* yang dikombinasikan dengan fungsi *IndexOf* untuk mengenali tanda tersebut).
Sedangkan di dalam prosedur *RefreshScreen*, pada awalnya mengatur property interval dari komponen *Timer* sehingga kecepatan dari label yang berjalan dapat tergantung dari setting yang telah dilakukan. Berikutnya, dilakukan pengaturan font (jenis dan ukuran), dan yang terakhir adalah melakukan penggantian teks yang telah diset sebelumnya. Dari semua pengaturan tersebut, pengambilan datanya menggunakan function *ambilkonfig*, sehingga antara function dan prosedur tersebut saling berkaitan satu sama lain.

4. Kemudian pindahkan event form ke event *KeyPress*, dan di dalam prosedur *MarqueeScreenSaver_KeyPress* ketikkan listing berikut :

```
If e.KeyChar = Chr(27) Then Close()
If e.KeyChar = Chr(32) Then
    MarqueeOption.ShowDialog()
    Me.Cursor.Show()
End If
```

Keterangan Listing :
Sepertihalnya pada contoh screensaver sebelumnya, listing di dalam sub *KeyPress* digunakan untuk mendeteksi penekanan tombol oleh pengguna. Penangkapan fungsi *e.KeyChar* akan mendeteksi penekanan tombol di keyboard berdasarkan kode ASCII yang dikenali. Kode ASCII 27 (Chr(27)) berarti pengguna sedang menekan tombol Escape, dan aplikasi akan dihentikan. Sedangkan pada saat pengguna menekan tombol spasi (kode ASCII 32), maka form untuk mengubah opsi screensaver akan ditampilkan ke layar.

5. Selanjutnya, pindahkan event form ke dalam event *MouseMove*, dan ketikkan listing berikutnya :

```
If Not aktif Then
    posisi = New Point(e.X, e.Y)
    aktif = True
Else
    If Math.Abs(e.X - posisi.X) > 15 Or _
        Math.Abs(e.Y - posisi.Y) > 15 Then
        Close()
    End If
End If
```

Keterangan Listing :
Prosedur ini juga sama dengan prosedur di contoh sebelumnya yaitu untuk mendeteksi pergerakan mouse (lihat contoh screensaver di sub bab sebelumnya.

Langkah penyelesaian untuk form yang kedua :

1. Kini berpindahlah ke form yang kedua, dengan melakukan dobel klik form kedua di dalam Solution Explorer.
2. Drag sebuah komponen *FontDialog* yang terletak di dalam tab *Dialog* pada Toolbox ke dalam form tersebut.

Komponen fontdialog merupakan komponen yang berfungsi untuk membukan sebuah kotak dialog pemilihan font dalam sebuah aplikasi. Daftar font yang muncul di dalam kotak dialog tersebut, nantinya akan bergantung kepada jenis font apa saja yang terinstalasi di dalam komputer yang sedang digunakan.

3. Dobel klik di form untuk mengetikkan listing yang berfungsi untuk menampilkan kembali kursor mouse (karena di form sebelumnya, kursor mouse dalam keadaan *hidden*) :

```
Me.Cursor.Show()
```

4. Berikutnya, deklarasikan variabel awal tepat di bawah *Public Class* …., dengan mengetikkan listing berikut :

```
Dim xFont, xsize As String
```

Keterangan Listing :
Dua variabel yang dideklarasikan di bagian paling atas berarti dapat digunakan di seluruh prosedur atau fungsi yang ada di dalam form. Variabel yang pertama yaitu *xFont* akan digunakan untuk menampung nama jenis huruf atau font yang akan digunakan di dalam screensaver. Sedangkan variabel kedua yaitu *xsize* digunakan untuk menyimpan ukuran font yang akan digunakan.

5. Kemudian dobel klik di button *Set Font* lalu ketikkan listing berikut di dalam *Button1_Click*

```
With FontDialog1
    If .ShowDialog = _
        Windows.Forms.DialogResult.OK Then
        xFont = .Font.FontFamily.Name
        xsize = .Font.Size
    End If
End With
```

Keterangan Listing :
Pada button atau tombol *Set Font* saat diklik akan menampilkan kotak dialog pemilihan font. Perlu diingat bahwa font yang akan muncul dalam kotak dialog tersebut adalah font yang terinstalasi

di dalam komputer masing-masing pengguna. Sehingga bisa terjadi keragaman font di setiap komputer yang berbeda.

6. Selanjutnya, dobel klik di button *Save Config* dan ketikkan listing ini :

```
Dim xspeed As String
For Each i As RadioButton In _
    Me.GroupBox1.Controls
    If i.Checked Then xspeed = i.Tag
Next
Dim teks As String = "speed=" & _
    xspeed & vbCrLf & _
    "font=" & IIf(xFont <> "", _
    xFont, "verdana") & vbCrLf & _
"fontsize=" & IIf(xsize <> "", xsize, "20") & _
  vbCrLf & "teks=" & IIf(TextBox1.Text <> "", _
    TextBox1.Text, "Learning By Sample")
My.Computer.FileSystem. _
    WriteAllText("marquee.cfg", teks, False)
Me.Cursor.Hide()
MarqueeScreenSaver.RefreshScreen()
Close()
```

Keterangan Listing :

Penyimpanan konfigurasi di dalam screensaver ini sebenarnya hanya melakukan proses penyimpanan pada sebuah file format teks dengan nama *marquee.cfg*. Terdapat empat konfigurasi yang disimpan yaitu kecepatan atau *speed* yang nantinya akan mengatur kecepatan pengaturan property interval dari timer.

Sedangkan konfigurasi kedua dan ketiga merupakan penyimpanan konfigurasi untuk jenis font serta ukuran font yang akan digunakan di dalam tulisan yang berjalan atau *marquee*. Kedua konfigurasi tersebut didapat dari kotak dialog font yang dieksekusi saat button *Set Font* diklik oleh pengguna.

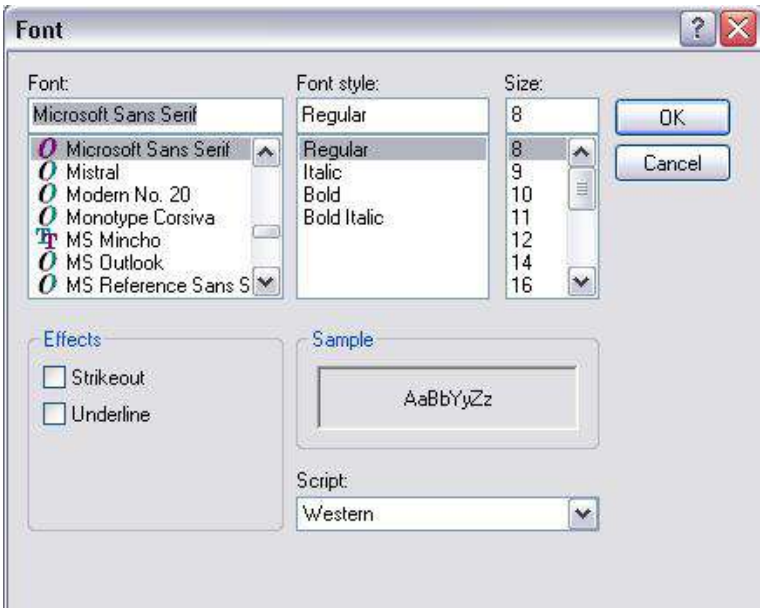
Dan untuk konfigurasi yang terakhir adalah konfigurasi tulisan yang akan ditampilkan oleh screensaver. Secara default, jika tulisan tidak diisi oleh pengguna, maka tulisan di dalam screensaver akan diisi dengan tulisan *Learning By Sample*.

7. Yang terakhir, dobel klik di button *Cancel* dan ketikkan listing berikut :

```
Close()
```

8. Kini, cobalah untuk melakukan eksekusi aplikasi tersebut dengan menekan tombol F5. Jika semua yang diketikkan telah benar, maka akan muncul sebuah layar dengan kalimat *Learning By Sample* yang berjalan dari kiri ke kanan layar dan kemudian akan melakukan scrolling berputar kembali pada saat tulisan tersebut telah sampai di ujung kanan layar. Kemudian, cobalah untuk menekan tombol spasi saat aplikasi berjalan, maka akan muncul form yang kedua yaitu untuk melakukan setting *Option* di screensaver tersebut.

9. Di dalam form untuk pemilihan option tersebut, cobalah untuk menekan button *Set Font* agar kotak dialog font muncul seperti pada gambar berikut :



10. Kemudian, dapat dicoba untuk menyimpan konfigurasi baru di dalam screensaver tersebut dengan menekan button *Save Config*.
11. Dan untuk melakukan pengecekan hasil konfigurasi yang baru, bisa dilihat di dalam folder *bin/debug* pda solution yang baru saja dibuat, lalu cari file dengan nama *marquee.cfg*. Selanjutnya buka file tersebut dengan aplikasi notepad, maka akan tampak isi file seperti pada contoh berikut (isi file bisa bervariasi, bergantung pada hasil setting konfigurasi terakhir) :

```
speed=10
font=verdana
fontsize=20
teks=Test Marquee Screensaver
```

12. Yang terakhir, lakukanlah instalasi screensaver seperti pada langkah instalasi yang dilakukan di contoh pertama. Dan kini Anda telah mendapatkan sebuah screensaver baru lengkap dengan sebuah form untuk melakukan penyimpanan konfigurasi didalamnya.

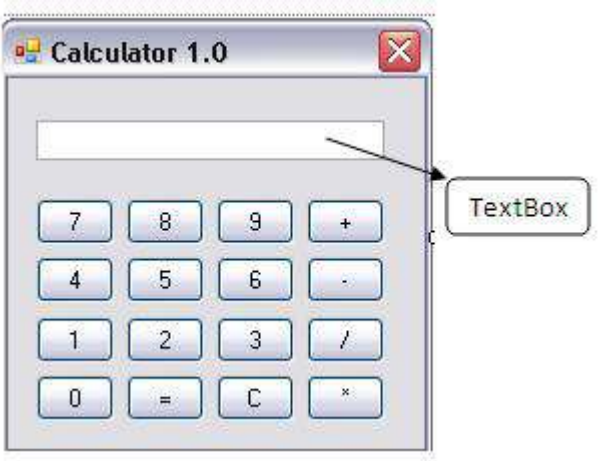
Saran Pengembangan :
Buatlah sebuah proses *deployment* yang nantinya dapat langsung melakukan instalasi screensaver tersebut ke dalam system Windows secara otomatis.

Contoh 3 : Kalkulator Sederhana

Level : Menengah
Tujuan :

1. Membuat aplikasi tiruan kalkulator dalam mode yang lebih sederhana
2. Mengaplikasikan penggunaan prosedur dan fungsi dalam sebuah aplikasi
3. Mengaplikasikan penggunaan prosedur dalam komponen secara dinamis
4. Menggunakan percabangan majemuk
5. Menggunakan perulangan *For Each* dalam sebuah form untuk iterasi komponen

Desain form awal :



Langkah penyelesaian :

1. Buat sebuah form yang didalamnya terdapat sebuah textbox dan enambelas button. Tiap button diberikan property *Text* seperti yang tertera dalam gambar. Untuk memudahkan pembuatan button tersebut, gunakan teknik *copy-paste* sehingga ukuran dari tiap button sama.

Agar button terlihat sama seperti desain form pada gambar, set property *TextAlign* pada button menjadi *MiddleCenter* dan property *Size* menjadi *39,23*. Sedangkan di komponen textbox,

set property ***TextAlign*** menjadi ***Right*** dan property ***ReadOnly*** menjadi ***True***

2. Selanjutnya, dobel klik pada form, dan didalam prosedur *Form_Load* ketikkan listing berikut :

```
For Each i As Object In Me.Controls
    If TypeOf i Is Button Then
        Dim  xButton  As  Button  =  CType(i,
Button)
        If IsNumeric(i.Text) Then
          AddHandler xButton.Click, _
            AddressOf klikAngka
        Else
            AddHandler xButton.Click, _
             AddressOf OperasiBilangan
        End If
    End If
Next
```

Keterangan Listing :

Pada saat form pertama kali dijalankan, maka tiap button yang ada di dalam kalkulator tersebut terbagi menjadi dua kelompok yaitu kelompok angka (button yang memiliki teks berupa angka) dan kelompok operasi bilangan (button yang berfungsi selain mengetikkan angka termasuk operasi penjumlahan, pengurangan, perkalian, pembagian, reset bilangan serta untuk mendapatkan hasil perhitungan).

Demi efisiensi, maka button yang ada di dalam form kalkulator tidak didobelklik satu per satu untuk mengetikkan prosedur yang ada di dalam tiap button. Sebagai gantinya, diterapkan konsep *dynamic control* dengan menggunakan perintah *AddHandler* yang berarti mengarahkan sebuah prosedur ke satu komponen tertentu secara dinamis.

Untuk mendapatkan komponen-komponen yang telah ditempatkan di dalam sebuah form dapat digunakan perintah iterasi (perulangan) *For Each* yang kemudian diikuti dengan variabel bertipe *Object* dan diulang di dalam sebuah form dengan melakukan iterasi di *Me.Controls*. Kata kunci *Me* dalam hal ini adalah form yang sedang aktif, sedangkan property *Controls* berarti bahwa iterasi tersebut akan melakukan pengecekan di tiap komponen yang ada dalam form tersebut.
Tetapi, karena komponen selain button yang ada di form tersebut, maka perlu dilakukan pengecekan terlebih dulu. Pengecekan diawali dengan perintah percabangan If yang akan dieksekusi jika tipe dari komponen adalah button (deklarasi variabel *xButton* yang bertipe button, sehingga memudahkan pada proses berikutnya).
Jika property *Text* pada button bertipe numerik (ditandai dengan percabangan *If Isnumeric*), maka button-button dalam kelompok tersebut akan diarahkan ke prosedur *KlikAngka* (yang akan dijelaskan di langkah berikutnya). Sedangkan jika tidak, maka akan diarahkan ke prosedur *OperasiBilangan* (juga akan dijelaskan di langkah selanjutnya). Pengarahan prosedur ke kelompok button tersebut dilakukan dengan menggunakan perintah *AddHandler* yang diikuti dengan method dari komponen yang akan diarahkan (dalam kasus ini method default dari button adalah *Click)* dan untuk mengarahkan prosedur digunakan perintah *AddressOf* yang kemudian diikuti dengan nama prosedur yang akan dieksekusi nantinya.

3. Kemudian, tepat di bawah deklarasi *Public Class…* ketikkan deklarasi variabel-variabel berikut ini :

```
Dim angka1, angka2 As Integer
Dim operasi As String
```

Keterangan Listing :
Dua variabel bertipe integer yang dideklarasikan tersebut akan digunakan pada saat perhitungan di dalam kalkulator. Sedangkan variabel *operasi* nantinya berfungsi untuk menyimpan jenis operasi bilangan yang akan diklik oleh pengguna.

4. Selanjutnya ketikkan prosedur *klikAngka* yang sebelumnya telah dideklarasikan pada saat *Form_Load*. Isi dari prosedur tersebut adalah :

```
Sub klikAngka(ByVal sender As Object, ByVal e
As EventArgs)
      Dim xBoleh As Boolean = True
      If TextBox1.Text = "" Then
          If sender.text = "0" Then
          MsgBox("Hanya bisa bilangan bulat !")
              xBoleh = False
          End If
      End If
If xBoleh Then TextBox1.Text &= sender.text
      If operasi <> "" Then _
            TextBox1.Text = sender.text
End Sub
```

Keterangan Listing :
Alur program dari pengetikan angka pada kalkulator sesungguhnya sangat sederhana. Hal yang pertama dilakukan adalah dengan melakukan pengecekan, apakah angka tersebut merupakan angka pertama dari textbox. Jika memang angka pertama, maka pengetikan angka akan dibatalkan karena

kalkulator di contoh ini dibatasi hanya untuk bilangan bulat (Anda bisa mengembangkan aplikasi ini untuk dapat melakukan perhitungan untuk angka desimal di kemudian hari). Jika ternyata bukan angka pertama, maka angka akan ditambahkan di belakang angka yang sudah ada.
Penggunaan kata kunci *sender* merupakan penggunaan obyek yang sedang aktif dieksekusi oleh prosedur tersebut. Sehingga button manapun yang dieksekusi oleh prosedur *klikAngka* akan dibaca sebagai sender. Dan untuk melakukan deteksi, kelompok button perlu ditandai dengan identitas tertentu, dalam hal ini identitas tersebut telah diperoleh dari property *Text* tiap button yang dipastikan berbeda untuk tiap angka.

5. Berikutnya, ketikkan prosedur *OperasiBilangan* yang berfungsi untuk menangani kelompok button yang memiliki property *Text* non angka. Prosedur ini memiliki tiga fungsi yaitu untuk membersihkan teks di dalam textbox (clear text), mengeksekusi hasil perhitungan serta melakukan perhitungan di dalam kalkulator.

```
Sub OperasiBilangan(ByVal sender As Object,
ByVal e As EventArgs)
Select Case sender.text
    Case "C"
        bersihTeks()
        TextBox1.Clear()
    Case "="
        Hitung()
    Case Else
        If TextBox1.Text = "" Then
            MsgBox("Belum ada angka pertama !")
```

```
        Else
            If angka1 = 0 Then
                angka1 = CInt(TextBox1.Text)
                operasi = sender.text
            Else
                Hitung()
                operasi = sender.text
                angka1 = CInt(TextBox1.Text)
            End If
        End If
End Select
End Sub
```

Keterangan Listing :
Pada prosedur tersebut terdapat implementasi *nested branching* atau percabangan bersarang (percabangan di dalam percabangan) yaitu adanya percabangan *if..then..else* di dalam percabangan *Select Case*. Hal tersebut terjadi di saat prosedur tidak melakukan aksi pembersihan textbox dan eksekusi akhir hasil perhitungan (ditandai dengan pemanggilan prosedur *Hitung*).
Pada saat proses perhitungan operasi bilangan, maka akan dilakukan pengecekan apakah operasi tersebut adalah operasi pertama atau lanjutan dari operasi bilangan yang sebelumnya (sebagai contoh adalah penjumlahan setelah pengurangan). Jika ya, maka akan dilakukan penyimpanan variabel berdasarkan tombol operasi bilangan yang ditekan oleh pengguna (dengan menggunakan pengenal *sender*). Dan jika tidak, maka prosedur hitung akan dieksekusi.

6. Prosedur terpenting di dalam aplikasi kalkulator sederhana ini adalah prosedur untuk melakukan perhitungan berdasarkan operasi matematika yang dipilih oleh pengguna.

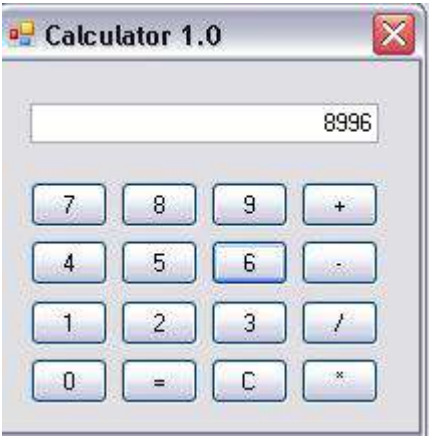
Sebelum percabangan dieksekusi, maka dilakukan terlebih dulu pengecekan terhadap angka-angka yang telah dimasukkan. Karena dalam kalkulator sederhana ini hanya memperbolehkan operasi matematika yang membutuhkan dua angka (Anda juga bisa mengembangkan kalkulator ini dengan operasi matematika yang hanya membutuhkan satu angka seperti kuadrat dan akar kuadrat), maka harus ada dua angka yang tersimpan dalam variabel awal yaitu *angka1* dan *angka2*.

```
Sub Hitung()
If angka1 <> 0 And operasi <> "" And
TextBox1.Text <> "" Then
    angka2 = CInt(TextBox1.Text)
    Dim xHasil As Integer
    Select Case operasi
        Case "+"
            xHasil = angka1 + angka2
        Case "-"
            xHasil = angka1 - angka2
        Case "*"
            xHasil = angka1 * angka2
        Case Else
            xHasil = angka1 / angka2
    End Select
    TextBox1.Text = xHasil
    bersihTeks()
End If
End Sub
```

7. Prosedur terakhir yang harus diketikkan adalah prosedur yang nantinya digunakan untuk membersihkan variabel yang telah digunakan dalam operasi matematika, sehingga dapat digunakan untuk operasi matematika berikutnya.

```
Sub bersihTeks()
      angka1 = 0
      angka2 = 0
      operasi = ""
End Sub
```

8. Contoh dari tampilan kalkulator sederhana yang telah selesai.



Saran Pengembangan :
Anda dapat menambahkan fungsi lain dari sebuah kalkulator standard seperti akar kuadat dan juga kuadrat.

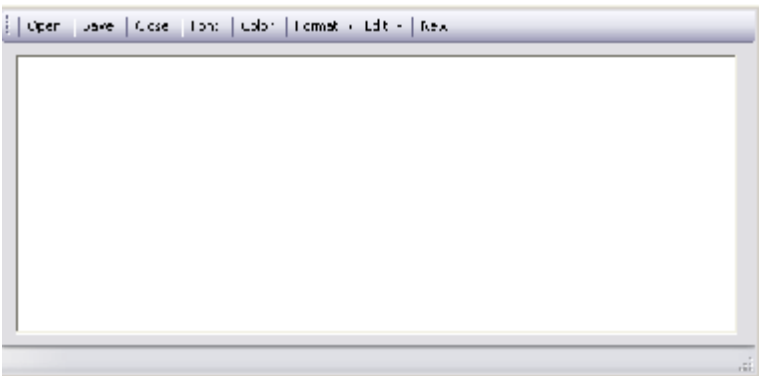
Contoh 4 : Editor Sederhana

Level : Menengah
Tujuan :

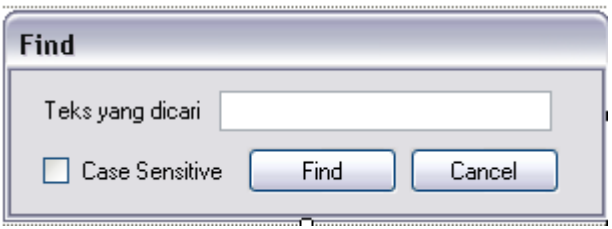
1. Membuat aplikasi editor *rich text format* sederhana
2. Mengaplikasikan teknik komponen *rich text box*
3. Mengaplikasikan teknik pemakaian kotak dialog dalam aplikasi (open, save,color dan font)
4. Menggunakan komponen *status strip* dan *tool strip*
5. Mengenal manipulasi string dalam komponen *rich text box*
6. Mengenal penggunaan *multiple handles* dalam prosedur.

Aplikasi terdiri dari dua form, yaitu form utama (frmUtama) dan form untuk pencarian kata (frmFind). Desain awal dari tiap form adalah :

1. Untuk frmUtama

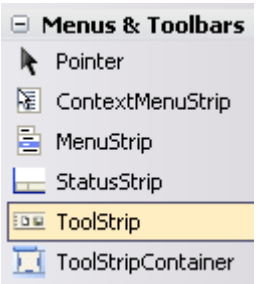


2. Desain frmFind



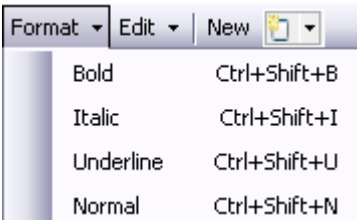
Langkah penyelesaian untuk form utama :

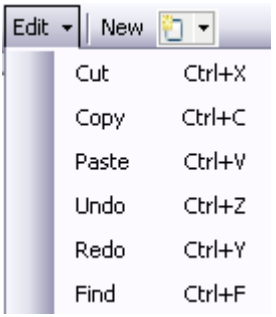
1. Pada form pertama (frmUtama), drag sebuah komponen *ToolStrip* ke dalam form dan isikan didalamnya item dengan struktur sebagai berikut :



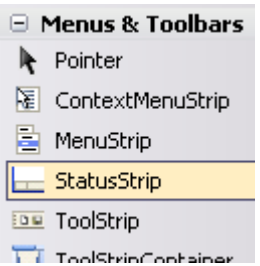
Komponen ToolStrip merupakan pengganti komponen Toolbar di .NET Framework versi sebelumnya. Dalam studi kasus ini, ToolStrip sengaja diisi dengan menggunakan mode teks bukan dengan mode icon. Hal ini sengaja dilakukan agar pada saat implementasi tidak terjebak pada kebingungan pemilihan icon. Tetapi jika Anda ingin mengubah ToolStrip ke mode icon, maka tinggal mengganti tiap item menjadi icon yang Anda sukai.

Struktur dari ToolStrip mirip pada gambar desain awal frmUtama, kecuali untuk menu *Format* dan menu *Edit* yang memiliki sub menu seperti pada gambar berikut :



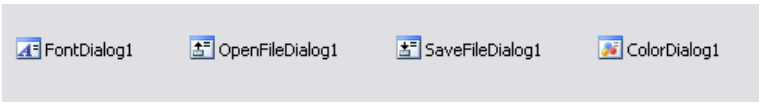


2. Selanjutnya drag sebuah komponen *StatusStrip* ke dalam form.



3. Untuk kepentingan penyimpanan dan pengeditan teks drag masing-masing komponen untuk kotak dialog yaitu komponen

: *FontDialog*, *ColorDialog. OpenFileDialog* dan *SaveDialog*.



Komponen yang terletak dalam grup *Dialog* pada saat proses desain form (design time) tidak akan terlihat. Tetapi untuk penggunaannya harus dipanggil pada saat form dieksekusi (runtime).

4. Kemudian untuk tempat pengeditan teks, drag sebuah komponen *RichTextBoxControl*

5. Selanjutnya, dobel klik pada bagian kosong formm dan di dalam prosedur *Form_Load* ketikkan listing berikut ini :

```
With StatusStrip1
    .Items.Add(Format(Now.Date, _
       "dd-MM-yyyy") & " | ")
    .Items.Add("Line 0 | ")
    .Items.Add("Char 0 | ")
    .Items.Add("Noname")
End With
Me.Text = JudulAplikasi
```

Keterangan Listing :

Pada saat form pertama kali mengalami proses loading, maka akan dilakukan setting pada komponen StatusSTrip. Pada .NET versi sebelumnya, komponen ini lebih dikenal dengan nama StatusBar yang memiliki sifat default *docking* ke bagian bawah form. Pada StatusStrip di dalam form tersebut, ditambahkan empat bagian atau item baru. Item yang pertama diisi dengan tanggal sistem berformat tanggal, bulan dan tahun (sesuai dengan format di Indonesia). Sedangkan item kedua dan ketiga nantinya akan berisi informasi mengenai jumlah baris dan jumlah karakter yang ada di dalam komponen RichTextBox (atau file teks yang sedang diedit). Dan item yang terakhir dalam StatusStrip nantinya akan diisikan dengan nama file teks yang sedang diedit (pada saat awal inisialisasi diberi keterangan sebagai *Noname* atau tanpa nama).

Baris terakhir dalam listing tersebut adalah melakukan setting terhadap *caption* dari form. Trik ini dapat Anda lihat di berbagai aplikasi Microsoft Office seperti Word dan Excel yang menampilkan nama file plus nama aplikasi di bagian paling atas. Variabel-variabel yang digunakan di dalam listing tersebut akan dijelaskan pada urutan berikutnya.

6. Di bagian paling atas dari listing program yaitu di bawah deklarasi *Public Class ….* Ketikkan deklarasi variabel-variabel berikut ini :

```
Dim JudulAplikasi As String = _
      "Editor Sederhana 1.0"
Dim Ekstension As String = _
      "RTF"
Dim filterFile As String = _
      "Rich Text Format|*.rtf"
Dim fileAktif As String
```

Keterangan Listing :
Dalam deklarasi awal variabel dengan jangkauan di dalam satu form tersebut, terdapat empat buah variabel yaitu :

a. *JudulAplikasi* yang digunakan untuk menampilkan caption pada form
b. *Ekstension* yang berfungsi untuk menentukan ekstensi file default pada saat kotak dialog open/save file teks
c. *filterFile* yang digunakan untuk filter otomatis pada saat kotak dialog open/save file teks
d. *fileAktif* yang nantinya akan diisi dengan file yang sedang diedit pada RichTextBox dan ditampilkan di StatusStrip.

7. Berikutnya ketikkan dua buah prosedur berikut yang nantinya akan digunakan di beberapa menu yang terdapat di dalam ToolStrip. Kedua prosedur tersebut adalah prosedur *SimpanFile* untuk melakukan penyimpanan file (temporer atau pada saat aplikasi ditutup) dan prosedur

refreshStatus yang dipanggil agar keterangan yang ada di dalam StatusStrip dapat terupdate sesuai dengan isi teks.

Prosedur refreshStatus

```
Sub refreshStatus()
With StatusStrip1
    .Items(1).Text = "Line(s) : " & _
        RichTextBox1.Lines.Count & " | "
    .Items(2).Text = "Char(s) : " & _
        RichTextBox1.Text.Count & " | "
    .Items(3).Text = _
        IIf(fileAktif <> "", fileAktif, _
            "Noname") & " | "
End With
End Sub
```

Keterangan Listing :
Secara sekilas, isi dari prosedur ini mirip dengan isi pada prosedur *Form_Load*. Perbedaan yang paling menyolok adalah isi dari tiap item di dalam StatusStrip diberikan percabangan *Iif* (inline If yang berarti percabangan dengan hanya satu kondisi sehingga bisa dijadikan lebih singkat dalam satu baris). Percabangan yang pertama melakukan perhitungan baris teks di dalam komponen RichTextBox, dan yang kedua melakukan perhitungan karakter di dalam komponen RichTextBox. Sedangkan percabangan terakhir melakukan pengecekan apakah variabel *fileAktif* memiliki isi atau tidak. Jika variabel tersebut memiliki nilai didalamnya, maka berarti file teks yang sedang diedit telah disimpan atau merupakan file teks yang telah ada sebelumnya.

Prosedur simpanFile

```
Sub simpanFile()
If fileAktif <> "" Then
    RichTextBox1.SaveFile(fileAktif)
    RichTextBox1.Modified = False
Else
    With SaveFileDialog1
        .Title = JudulAplikasi
        .DefaultExt = Ekstension
        .Filter = filterFile
        If .ShowDialog() = _
            Windows.Forms.DialogResult.OK Then
            If .FileName <> "" Then
            RichTextBox1.SaveFile(.FileName, _
                RichTextBoxStreamType.RichText)
                fileAktif = .FileName
                RichTextBox1.Modified = False
            End If
        End If
    End With
End If
refreshStatus()
End Sub
```

Keterangan Listing :

Pada saat proses penyimpanan file, terlebih dulu dicek apakah teks yang ada di dalam RichTextBox telah disimpan sebelumnya. Jika memang sudah pernah disimpan sebelumnya, maka komponen RichTextBox akan diperintahkan untuk melakukan penyimpanan ulang.

Tetapi, jika teks di dalam RichTextBox belum pernah disimpan, maka kotak dialog penyimpanan (SaveFileDialog) akan dipanggil terlebih dulu. Jika memang pengguna melakukan proses penyimpanan (dengan menekan tombol OK/Save pada kotak

dialog), maka penyimpanan yang sesungguhnya dieksekusi dan variabel *fileAktif* akan diisi dengan nama file yang telah diisikan. Bagian terakhir dalam prosedur ini adalah memanggil prosedur *refreshStatus* agar informasi yang tertera di dalam StatusStrip dapat terupdate sesuai dengan apa yang telah dilakukan oleh pengguna pada proses penyimpanan.

8. Lalu dobel klik pada komponen RichTextBox, dan didalam prosedur *RichTextBox1_TextChanged* ketikkan listing berikut untuk melakukan refresh teks yang terdapat di dalam StatusStrip.

```
refreshStatus()
```

9. Sedangkan untuk eksekusi tiap item di dalam ToolStrip, ikuti langkah-langkah berikut (dobel klik di tiap item dalam ToolStrip untuk mengetikkan listing programnya) :

Item Open

```
With OpenFileDialog1
    .DefaultExt = Ekstension
    .Title = JudulAplikasi
    .Filter = filterFile
    .FileName = String.Empty
    If .ShowDialog() = _
        Windows.Forms.DialogResult.OK Then
        If .FileName <> "" Then
            RichTextBox1.LoadFile(.FileName, _
            RichTextBoxStreamType.RichText)
```

```
            RichTextBox1.Modified = False
            fileAktif = .FileName
            refreshStatus()
        End If
    End If
End With
```

Keterangan Listing :

Untuk membuka file teks, maka secara default akan dipanggil kotak dialog open file (OpenFileDialog). Setting properti di dalam komponen tersebut yang diatur di dalam listing adalah property *DefaultExt* dan *FilterFile* yang menyatakan ekstensi file default yang boleh dibuka untuk aplikasi editor sederhana. Kemudian property *Title* yang diisikan dengan variabel *JudulAplikasi* sehingga menjadi standard di dalam program. Dan yang terakhir adalah property *FileName* yang langsung diset menjadi kosong (String.Empty) sehingga pada saat kotak dialog pertama kali dibuka, maka pengguna bebas untuk memilih file secara langsung.

Eksekusi *loading* file ke dalam komponen RichTextBox baru akan dilaksanakan jika telah berhasil melewati beberapa pengecekan yaitu pada saat nama file telah memiliki nilai (berarti bahwa pengguna telah benar-benar akan membuka file teks sesuai dengan filter yang telah ditetapkan) dan apakah pengguna telah menekan tombol OK/Open pada kotak dialog yang tersedia. Contoh dari OpenFileDialog tampak pada gambar berikut ini :

Item Save

```
simpanFile()
```

Keterangan Listing :
Pada item untuk melakuan penyimpanan maka hanya diperlukan satu baris listing program untuk memanggil ulang prosedur *SimpanFile* (lihat lagi di nomor penyelesaian sebelumnya).

Item Close

```
If RichTextBox1.Modified = False Then
    Close()
Else
    If MessageBox.Show("Simpan file dulu ?", _
```

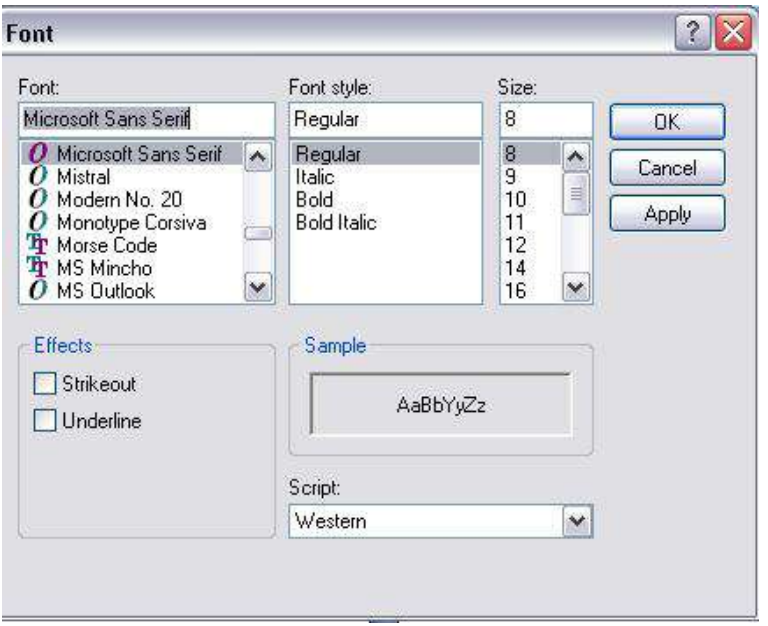
```
     "Konfirmasi", MessageBoxButtons.YesNo) = _
        Windows.Forms.DialogResult.Yes Then
        simpanFile()
        Close()
    Else
        Close()
    End If
End If
```

Keterangan Listing :
Sesungguhnya apa yang dilakukan pada saat aplikasi ditutup adalah menampilkan konfirmasi jika ternyata file teks yang sedang diedit di dalam RichTextBox belum disimpan oleh pengguna. Jika memang teks tersebut belum disimpan, maka pengguna akan dipaksa untuk menjawab kotak konfirmasi untuk melakukan penyimpanan file. Selanjutnya akan dipanggil kembali prosedur untuk melakukan penyimpanan file yaitu prosedur *SimpanFile*.

Item Font

```
With FontDialog1
    If IsNothing(RichTextBox1. _
                SelectionFont) Then
        .Font = Nothing
    Else
        .Font = RichTextBox1.SelectionFont
    End If
    .ShowApply = True
    If .ShowDialog() = _
        Windows.Forms.DialogResult.OK Then
        RichTextBox1.SelectionFont = .Font
    End If
End With
```

Keterangan Listing :
Pada menu untuk pemilihan font, maka akan dieksekusi kotak dialog pemilihan font (FontDialog). Kotak dialog font dapat digunakan oleh pengguna untuk memilih style huruf/font yang diinginkan untuk teks, didalamnya termasuk jenis font (jenis font akan berbeda di tiap komputer bergantung kepada font yang ada dalam komputer tersebut) dan juga style lainnya seperti ukuran hingga style lain (bold/tebal, italic/cetak miring dan strike/cetak dengan coret). Contoh dari hasil implementasi FontDialog tampak pada gambar berikut ini :

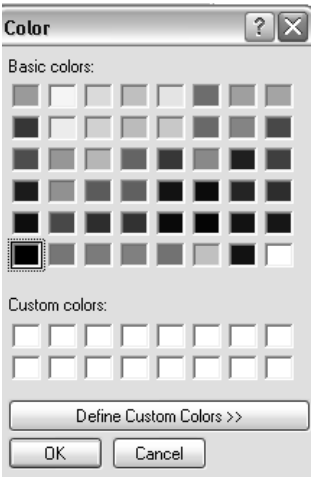


Item Color

```
With ColorDialog1
    .Color = RichTextBox1.SelectionColor
    If .ShowDialog() = _
```

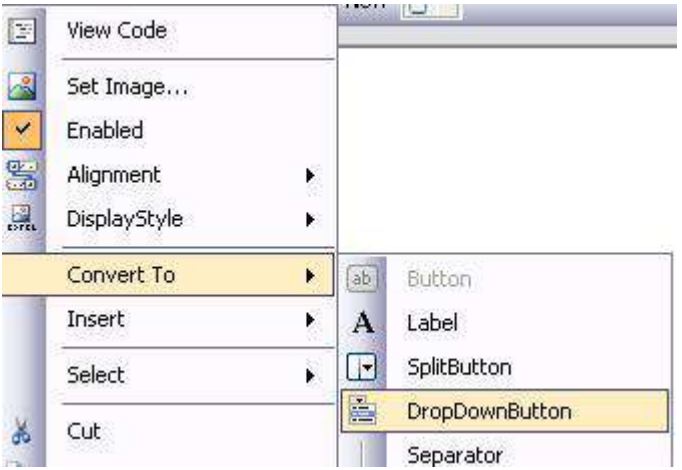
```
        Windows.Forms.DialogResult.OK Then
        RichTextBox1.SelectionColor = .Color
    End If
End With
```

Keterangan Listing :
Serupa dengan implementasi untuk menu pemilihan font, pemilihan warna atau color juga hanya melakukan eksekusi untuk membuka kotak dialog pemilihan warna atau ColorDialog. Contoh tampilan dari ColorDialog adalah sebagai berikut :

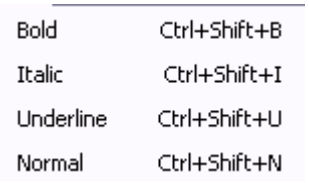


Item Format

Di dalam menu Format terdapat empat sub item menu yaitu untuk menu Bold, Italic, Underline dan Normal. Untuk membuat agar menu Format dapat memiliki sub item maka klik kanan pada menu Format dan pilih sub menu *Convert* kemudian ganti mode menjadi *DropDownButton*.



Setelah selesai melakukan konversi ke hal DropDownbutton, kemudian tambahkan empat sub item menu yang telah disebutkan sebelumnya. Pada saat menambahkan sub item menu usahakan agar memberi nilai pada property *Name* sesuai dengan judul menu agar lebih mempermudah langkah berikutnya, misal : untuk sub item menu *Bold*, maka property *Name* juga berisi *Bold*.



Pada masing-masing sub item menu terdapat deklarasi shortcut keyboard yang telah ditetapkan. Deklarasi shortcut tersebut dapat diperoleh dari property *ShortcutKeys*. Salah satu hal penting yang harus dihindari pada saat melakukan pembuatan shortcut adalah untuk tidak membuat shortcut yang sama dengan *common shortcut* atau shortcut umum yang telah ada di sistem operasi agar nantinya tidak rancu dengan shortcut yang asli, misal : shortcut F1 yang di dalam Windows merupakan default shortcut untuk menampilkan bantuan (Help).

Selanjutnya, dobel klik pada sub item menu *Bold* dan tepat di belakang deklarasi *Handles* ketikkan baris berikut :

```
BoldToolStripMenuItem.Click, _
      ItalicToolStripMenuItem.Click, _
      UnderlineToolStripMenuItem.Click, _
      NormalToolStripMenuItem.Click
```

Keterangan Listing :
Dengan menambahkan event baru dari komponen pada kata kunci setelah *Handles* maka berarti bahwa event dari komponen lain tersebut juga akan mengacu ke prosedur yang sama. Hal ini seringkali dilakukan apabila dari beberapa komponen sesungguhnya melakukan beberapa hal yang dianggap mirip tetapi dapat dibedakan satu sama lain dengan identitas tertentu. Identitas pembeda tersebut dapat dilihat pada listing berikutnya.

Kemudian di dalam prosedur tersebut ketikkan listing berikut :

```
With RichTextBox1
    Select Case sender.text
        Case "Bold"
            .SelectionFont = _
            New Drawing.Font _
            (.SelectionFont, FontStyle.Bold)
        Case "Italic"
            .SelectionFont = _
            New Drawing.Font _
            (.SelectionFont, FontStyle.Italic)
        Case "Underline"
            .SelectionFont = _
            New Drawing.Font _
          (.SelectionFont, FontStyle.Underline)
        Case "Normal"
            .SelectionFont = _
            New Drawing.Font _
            (.SelectionFont, FontStyle.Regular)
    End Select
End With
```

Keterangan Listing :

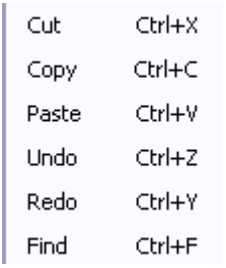
Karena di dalam prosedur ini langsung menangani empat sub item menu sekaligus, maka tiap sub item menu perlu diberi identitas yang jelas agar pada saat eksekusi dapat dibedakan satu dengan lainnya. Dalam prosedur ini identitas yang diambil adalah property *Text* dari tiap sub item menu. Dan untuk mengambil property tersebut, digunakan parameter *sender* yang berarti akan mengikuti tiap obyek yang sedang aktif dieksekusi (dalam hal ini obyek adalah tiap sub item menu yang termasuk di dalam *Handles*).

Setelah diidentifikasi tiap sub item menu yang sedang aktif, maka selanjutnya akan diatur property dari RichTextBox, agar

teks yang sedang dipilih (diblok) berubah menjadi style font yang diinginkan yaitu bold, italic, underline atau kembali ke style normal.

Item Edit

Di dalam menu Edit, sama halnya dengan menu Format, juga harus dikonversi terlebih dulu ke bentuk DropDownButton. Dan sama juga dengan langkah di sub item menu sebelumnya, tambahkan sub item baru untuk item Edit seperti tampak pada gambar berikut :



Untuk sub item menu *Cut, Copy, Paste, Undo* dan *Redo* merupakan operasi standard dalam pengolahan teks, sehingga nantinya akan dieksekusi oleh satu prosedur (lihat lagi teknik menggunakan *multiple Handles* di item menu *Format*). Sedangkan untuk sub item menu *Find* akan membukan form yang kedua yaitu *frmFind*.

Kini dobel klik di sub item menu *Cut*, dan tepat di belakang kata kunci *Handles,* ketikkan tambahan event berikut ini :

```
CutToolStripMenuItem.Click, _
    CopyToolStripMenuItem.Click, _
    PasteToolStripMenuItem.Click, _
    UndoToolStripMenuItem.Click, _
    RedoToolStripMenuItem.Click
```

Selanjutnya, di dalam prosedur tersebut ketikkan listing berikut ini :

```
Try
    With RichTextBox1
        Select Case sender.text
            Case "Cut"
                .Cut()
            Case "Copy"
                .Copy()
            Case "Paste"
                .Paste()
            Case "Undo"
                .Undo()
            Case "Redo"
                .Redo()
        End Select
    End With
Catch ex As Exception
    MsgBox("Error !")
End Try
```

Keterangan Listing :
Seperti telah disebutkan sebelumnya, bahwa operasi *Cut, Copy, Paste, Undo* dan *Redo* merupakan operasi standard dalam

pengolahan teks sehingga komponen RichTextBox memiliki method default untuk menangani operasi tersebut.

Berikutnya untuk sub item menu *Find*, dobel klik dan di dalam prosedur tersebut ketikkan listing berikut ini untuk membuka form yang kedua yaitu *frmFind* :

```
frmFind.ShowDialog()
```

Item New

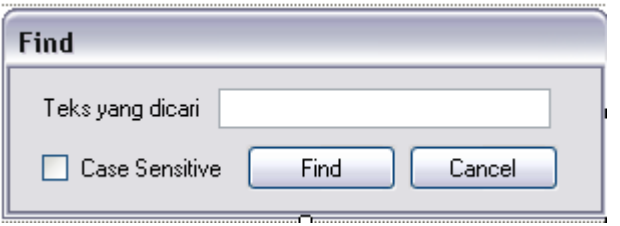
Item menu terakhir yang didobel klik adalah item menu *New* yang nantinya akan digunakan untuk membuat file teks baru.

```
If RichTextBox1.Modified Then
    If MessageBox.Show("Simpan file dulu ?", _
     "Konfirmasi", MessageBoxButtons.YesNo) = _
        Windows.Forms.DialogResult.Yes Then
        simpanFile()
    End If
End If
RichTextBox1.Clear()
fileAktif = ""
refreshStatus()
```

Keterangan Listing :
Pada prosedur ini, pembuatan file teks baru akan melakukan pengecekan jika ternyata terdapat file teks lain yang sedang diedit dan dalam keadaan belum tersimpan. Jika memang kondisi tersebut terjadi, maka akan dieksekusi terlebih dulu kotak dialog konfirmasi dan prosedur untuk melakukan penyimpanan

file teks (jika pengguna ingin menyimpan terlebih dulu, mirip dengan konfirmasi saat berada di dalam Microsoft Office).
Jika kondisi tersebut telah terlampaui, maka keadaan aplikasi akan direset ke keadaan awal yaitu membersihkan kembali isi dari RichTextBox dan mengosongkan variabel *fileAktif* serta melakukan refresh teks di dalam StatusStrip.

Langkah penyelesaian untuk form pencarian kata atau *frmFind* :



1. Dobel klik pada button *Find* lalu ketikkan listing berikut ini :

```
If Trim(TextBox1.Text) <> "" Then
    With frmUtama.RichTextBox1
        .Find(TextBox1.Text, _
              IIf(CheckBox1.Checked, _
                  RichTextBoxFinds.MatchCase, _
                  RichTextBoxFinds.None))
     If .Text.IndexOf(TextBox1.Text, 0, 0) Then
          MessageBox.Show("Tidak ditemukan !")
        End If
    End With
Else
    MessageBox.Show _
        ("Tidak ada kata yang dicari !")
End If
Close()
```

Keterangan Listing :
Pada form pencarian kata di dalam RichTextBox, dimanfaatkan method *Find* yang secara default akan melakukan pencarian kata di dalam RichTextBox dengan parameter yang telah disediakan. Pengecekan kondisi dilakukan jika ternyata teks yang telah diketikkan di dalam TextBox ternyata tidak ditemukan, maka akan diberikan pesan dalam kotak dialog. Dan secara default, jika teks ditemukan maka form ini akan ditutup dan di dalam komponen RichTextBox di form utama, kursor akan langsung menuju ke teks yang dicari.

2. Kemudian di button *Cancel* dobel klik juga dan ketikkan listing berikut untuk menutup form tanpa melakukan proses pencarian kata :

```
Close()
```

Kini jalankan aplikasi tersebut, dan Anda bisa mencoba untuk melakukan edit terhadap file teks tertentu.

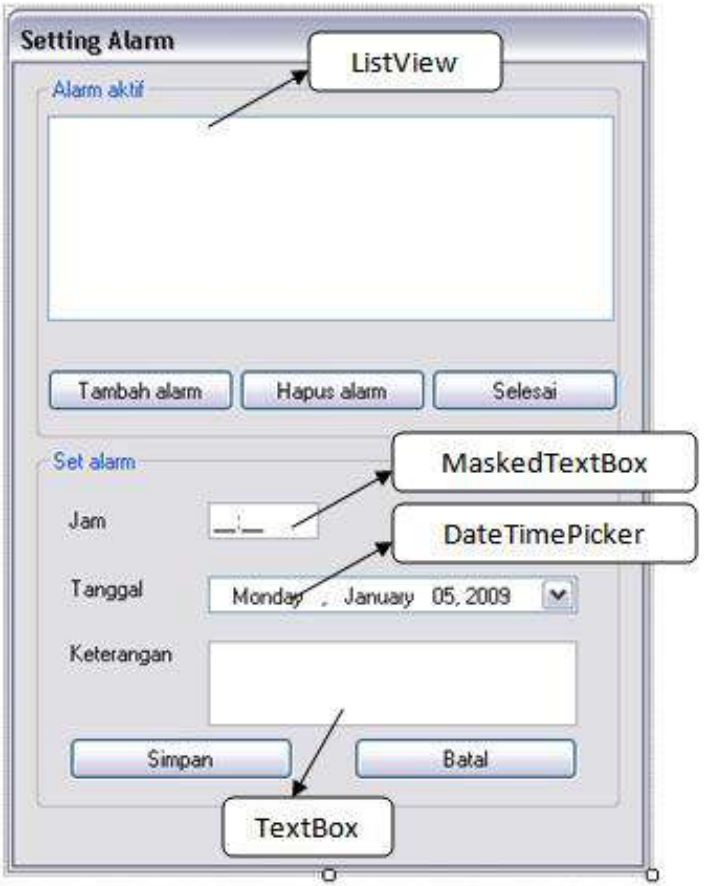
Saran Pengembangan :
Anda dapat menambahkan fasilitas untuk mencetak di dalam aplikasi tersebut.

Contoh 5 : Alarm

Level : Menengah
Tujuan :

1. Membuat aplikasi *alarm notification* yang mampu melakukan *alert* atau peringatan pada waktu tertentu yang datanya telah disimpan dalam sebuah file teks
2. Menggunakan komponen *NotifyIcon* dalam sebuah aplikasi model *stay residence*
3. Mengenal manipulasi file teks secara sekuensial
4. Mengenal penggunaan komponen *ListView*
5. Mengenal penggunaan komponen *MaskedTextBox*
6. Mengenal penggunaan komponen *GroupBox*
7. Membunyikan file format *.wav* dalam sebuah aplikasi.

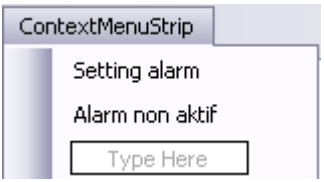
Desain awal :


Setting Alarm
ListView
Alarm aktif
Tambah alarm
Hapus alarm
Selesai
Set alarm
MaskedTextBox
Jam
DateTimePicker
Tanggal
Monday , January 05, 2009
Keterangan
Simpan
Batal
TextBox


NotifyIcon1
ContextMenuStrip1
Timer1

Langkah penyelesaian :

1. Klik pada komponen ContextMenuStrip, kemudian tambahkan sub item menu berikut ini :



ContextMenuStrip atau menu yang akan muncul pada saat pengguna melakukan klik kanan pada mouse, nantinya akan muncul di saat NotifyIcon diklik oleh pengguna. NotifyIcon merupakan icon kecuil yang muncul di pojok kiri bawah toolbar Windows.

2. Selanjutnya, dobel klik pada sub item menu *Setting Alarm* dan ketikkan listing berikut ini :

```
Me.WindowState = _
    FormWindowState.Normal
```

Keterangan Listing :
Pada saat sub item menu *Setting Alarm* diklik, maka form seolah-olah diaktifkan oleh aplikasi. Meski sesungguhnya proses aktifasi tersebut hanyalah mengembalikan form ke ukuran yang asli.

3. Lalu, dobel klik di sub item menu *Alarm non aktif* dan ketikkan listing berikut didalamnya :

```
Close()
```

Keterangan Listing :
Karena aplikasi ini meniru perilaku aplikasi TSR (Terminate and Stay Residence atau aplikasi yang dijalankan sekali dan kemudian tetap ada di sistem operasi hingga proses shutdown), maka untuk menonaktifkan aplikasi berarti juga menutup aplikasi itu sendiri.

4. Kemudian, klik pada form dan atur property form sebagai berikut :

ShowInTaskbar	**False**
Size	**348, 467**
SizeGripStyle	Auto
StartPosition	**CenterScreen**
Tag	
Text	**Setting Alarm**
TopMost	False
TransparencyKe	
UseWaitCursor	False
WindowState	**Minimized**

Property yang diatur di dalam form (dan terpenting dilakukan agar komponen NotifyIcon dapat aktif) adalah property *WindowState*. Dalam property ini wajib diset menjadi *Minimized* agar NotifyIcon dapat aktif pada saat aplikasi pertama kali dijalankan. Selain itu, property *ShowInTaskbar* juga harus diberi nilai

false agar form yang sedang diminimize tidak terlihat lagi.

5. Berikutnya, klik pada komponen Timer dan atur property sebagai berikut :

Enabled	**True**
GenerateMember	True
Interval	**1000**

6. Setelah itu, klik pada komponen NotifyIcon dan atur property seperti pada gambar berikut :

BalloonTipIcon	**Info**
BalloonTipText	**Alarm 1.0 - Klik kanan untuk menampilkan menu**
BalloonTipTitle	**Setting alarm**
⊞ ContextMenuStrip	**ContextMenuStrip1**
GenerateMember	True
⊞ Icon	**(Icon)**
Modifiers	**Friend**
Tag	
Text	**Alarm**
Visible	**True**

Property terpenting yang harus disetting dalam komponen NotifyIcon adalah property *Icon* yang harus diisikan dengan file dengan ekstensi *.ico* (Anda bisa mencari file tersebut di beberapa sub folder yang ada di dalam folder Windows). Jika property ini tidak diisikan, maka NotifyIcon tidak akan muncul di dalam StatusBar. Sedangkan property *Balloon* (baik dengan akhiran *TipIcon*, *TipText* maupun *TipTitle*) berfungsi sebagai penampil keterangan

di saat pengguna mengarahkan kursor mouse di icon.

7. Berikutnya, klik di komponen MaskedTextBox, dan isikan nilai *90:00* di dalam property *Mask*. Ini berarti bahwa di dalam MaskedTextBox akan menerima format jam dengan format *hh:mm* atau jam dan menit.
8. Kemudian dobel klik pada form, dan di dalam prosedur *Form_Load* ketikkan listing berikut :

```
With ListView1
    .View = View.Details
    .Columns.Add("Jam")
    .Columns.Add("Tanggal")
    .Columns.Add _
        ("Keterangan", 200)
End With
bacaData()
```

Keterangan Listing :
Pada saat form pertama kali dibuka, maka komponen ListView diberikan tiga kolom baru yang masing-masing untuk mengisi data *jam, tanggal* dan *keterangan* dari alarm. ListView juga diset menjadi mode *Details* agar dapat menampilkan kolom-kolom tersebut, sebab ListView memiliki beberapa mode tampilan antara lain, *thumbnail* dan *icon*.
Selanjutnya, prosedur untuk membaca data teks yaitu prosedur *bacaData* dieksekusi sebagai inisialisasi awal. Isi dari prosedur tersebut diterangkan di langkah berikutnya.

9. Kini, ketikkan prosedur untuk melakukan pembacaan file teks untuk data alarm dengan nama *bacaData*.

```
Sub bacaData()
If IO.File.Exists("alarmdata.dat") Then
    Dim xdata() As String = _
        Split(IO.File.ReadAllText _
            ("alarmdata.dat"), vbCrLf)
    ListView1.Items.Clear()
    For i As Integer = 0 To xdata.Length - 1
        Dim xdetail() As String = _
            Split(xdata(i), ",")
        With ListView1
            .Items.Add(xdetail(0))
            .Items(i).SubItems. _
                Add(xdetail(1))
            .Items(i).SubItems. _
                Add(xdetail(2))
        End With
    Next
End If
End Sub
```

Keterangan Listing :

File teks yang akan digunakan sebagai tempat penyimpanan data diberi nama *alarmdata.dat* dan secara default disimpan di dalam folder yang sama dengan folder aplikasi. Prosedur ini hanya dieksekusi jika file teks telah terbentuk sebelumnya, tetapi jika belum ada file teks (belum ada data alarm) maka isi dari prosedur ini akan diabaikan.

Pada saat file teks telah terbentuk, maka file teks akan dipecah berdasarkan tiap baris (ditandai dengan parameter *vbCrLf* pada fungsi *Split*). Selanjutnya dari tiap data yang dianggap sebagai

record tersebut dipecah lagi berdasarkan tanda koma (,) dan dimasukkan ke dalam ListView.

10. Selanjutnya, dobel klik pada button *Tambah Alarm* dan ketikkan listing berikut ini didalamnya :

```
GroupBox2.Enabled = True
GroupBox1.Enabled = False
```

Keterangan Listing :
Pada saat proses untuk melakukan penambahan data alarm baru, maka hal yang perlu dilakukan hanyalah melakukan aktifasi pada groupbox bagian bawah (Groupbox2), sehingga pengguna dapat melakukan proses entri data alarm baru.

11. Kemudian di button *Hapus Alarm* dobel klik juga dan ketikkan listing berikut :

```
With ListView1
    If .Items.Count > 0 Then
        Dim xindex As Integer = _
            .SelectedItems(0).Index
        Dim xdata() As String = _
            Split(IO.File.ReadAllText _
                  ("alarmdata.dat"), vbCrLf)
        Dim xTemp As String
       For i As Integer = 0 To xdata.Length - 1
            If i <> xindex Then
                xTemp &= xdata(i) & _
                IIf(i <> xdata.Length - 1, _
                    vbCrLf, "")
            End If
        Next
        .SelectedItems(0).Remove()
```

```
        IO.File.WriteAllText _
            ("alarmdata.dat", xTemp)
    Else
        MsgBox _
        ("Tidak ada alarm yang bisa dihapus !")
    End If
End With
```

Keterangan Listing :

Terdapat dua proses di dalam proses penghapusan data alarm yakni pembacaan data pada file teks secara sekuensial (ditandai dengan proses perulangan untuk mencari data yang diklik pada ListView) dan proses penghapusan data itu sendiri. Isi dari prosedur ini hanya akan dieksekusi pada saat ListView memilik data didalamnya.

Teknik proses penghapusan di dalam file teks hanyalah membaca seluruh teks, menghapus bagian yang diinginkan dan menulis ulang kembali teks yang telah dihapus tersebut. Proses ini memakan waktu yang lama apabila data di dalam file teks dianggap terlalu besar (misalkan lebih dari 1000 data/baris), karenanya di dalam implementasi yang sesungguhnya sangat tidak disarankan untuk memakai file teks pada data yang cukup besar.

12. Setelah itu, dobel klik pada button *Batal* di Groupbox bagian bawah dan ketikkan listing berikut ini :

```
GroupBox1.Enabled = True
GroupBox2.Enabled = False
```

Keterangan Listing :
Isi prosedur di button *Batal* sesungguhnya hanya kebalikan dari isi di dalam button *Tambah Alarm* yaitu untuk mengembalikan status GroupBox ke posisi awal.

13. Berikutnya, dobel klik pada button *Simpan* dan ketikkan listing berikutnya :

```
Dim xdata As String
Dim xpath As String = "alarmdata.dat"
Dim xbaris As Boolean = IO.File.Exists(xpath)
xdata = IIf(xbaris, vbCrLf, "") & _
    MaskedTextBox1.Text & _
    "," & Format(DateTimePicker1.Value, _
                 "dd-MM-yyyy") & _
    "," & TextBox1.Text
IO.File.AppendAllText(xpath, xdata)
bacaData()
Button5_Click(sender, e)
```

Keterangan Listing :
Pada prosedur ini hal yang pertama kali dilakukan di dalam proses adalah melakukan pengecekan apakah sebelumnya telah terbentuk file teks *alarmdata.dat*. Jika file teks belum ada maka isi dari data yang diinputkan oleh pengguna langsung diisikan ke dalam sebuah variabel string (xdata). Selanjutnya, variabel *xdata* tersebut ditambahkan ke dalam file teks dengan perintah *AppendAllText*.

14. Prosedur terakhir yang harus diketikkan adalah prosedur inti untuk mengaktifkan alarm. Prosedur ini diketikkan di dalam *Timer1_Tick* dengan jalan melakukan dobel klik pada

komponen Timer dan mengetikkan listing berikut ini :

```
If IO.File.Exists("alarmdata.dat") Then
    Dim xdata() As String = _
        Split(IO.File.ReadAllText _
              ("alarmdata.dat"), vbCrLf)
    For i As Integer = 0 To xdata.Length - 1
        Dim xdetail() As String = _
            Split(xdata(i), ",")
        If Format(Now.Date, _
                "dd-MM-yyyy") = xdetail(1) Then
            If Format(Now, "hh:mm") = _
                 xdetail(0) And _
                 Format(Now, "ss") = "00" Then
           My.Computer.Audio.PlaySystemSound _
                  (Media.SystemSounds.Asterisk)
                MessageBox.Show(xdetail(2), _
                    "Reminder Alert")
            End If
        End If
    Next
End If
```

Keterangan Listing :

Karena property interval dari Timer telah diset menjadi 1000 (atau sama dengan 1 detik), maka isi dari listing yang ada di dalam *Timer1_Tick* akan dieksekusi tiap detik selama aplikasi belum ditutup oleh pengguna. Dalam listing tersebut, yang dilakukan adalah membaca data dari file teks yang kemudian dicek dengan tanggal dan jam dari sistem. Selanjutnya, jika data jam alarm ternyata sama dengan jam sistem, maka akan dimunculkan sebuah kotak pesan dengan isi yang sesuai di keterangan, serta membunyikan file system audio jenis *asterisk*.

Anda juga bisa mengganti file audio dengan jenis yang lain sesuai dengan keinginan. Tetapi, perintah *PlaySistemSound*

hanya berfungsi untuk memainkan file audio yang menjadi default dari Windows, sehingga bukan dari file audio lain, misalkan format MP3.

15. Saat mencoba aplikasi yang telah dijalankan, maka tidak akan muncul sebuah form secara default, tetapi hanya sebuah icon di dalam system tray. Langkah selanjutnya adalah mencoba mengisikan data alarm (dengan interval yang tidak terlalu lama dari tanggal sistem), kemudian menunggu saat alarm dibunyikan.

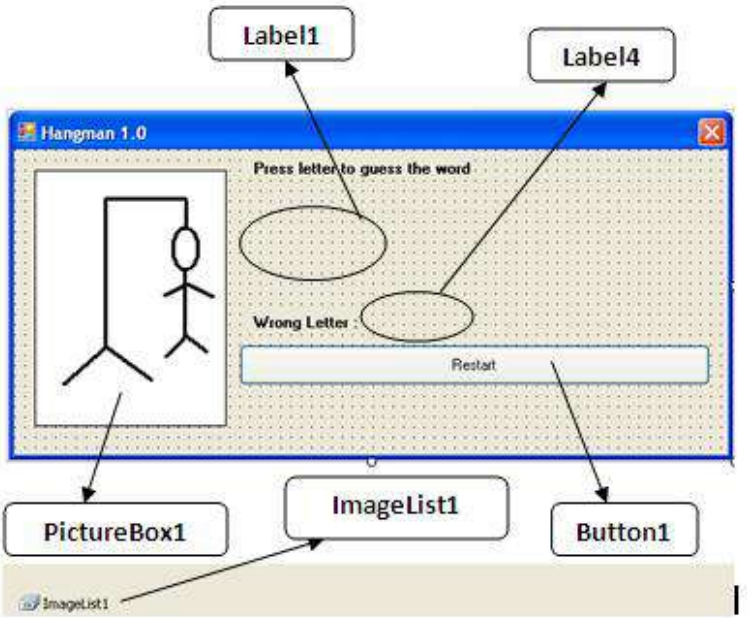
Saran Pengembangan :
Anda dapat menambahkan fasilitas agar aplikasi alarm ini dapat langsung berjalan saat Windows pertama kali melakukan proses booting.

Contoh 6 : Game Hangman

Level : Menengah
Tujuan :

1. Membuat aplikasi game sederhana yang mendemonstrasikan penggunaan image list yang memiliki rangkaian gambar dan deteksi penekanan alfabet di dalam sebuah form
2. Mengenal penggunaan ImageList
3. Mengenal manipulasi array
4. Mengenal deteksi alfabet dari kode ASCII
5. Mengenal kombinasi penggunaan PictureBox dan ImageList
6. Menggunakan file teks sebagai soal penyelesaian game

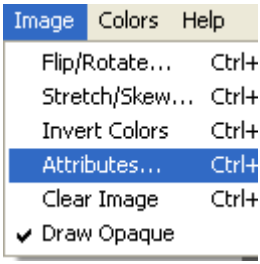
Desain awal :

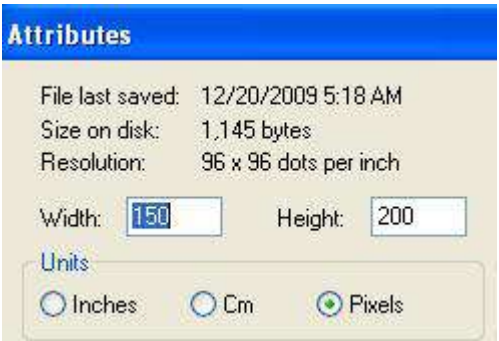


Langkah penyelesaian :

1. Sebelum membuat desain form, terlebih dulu buatlah lima buah file yang nantinya akan membentuk gambar *hangman* (orang yang digantung di tiang). Lima buah file tersebut, masing-masing diberi nama secara berurutan dengan ekstensi *.png* (juga bisa dengan format lain seperti *.jpg* atau *.bmp*). Pembuatan file gambar tersebut dalam contoh ini hanya dibuat

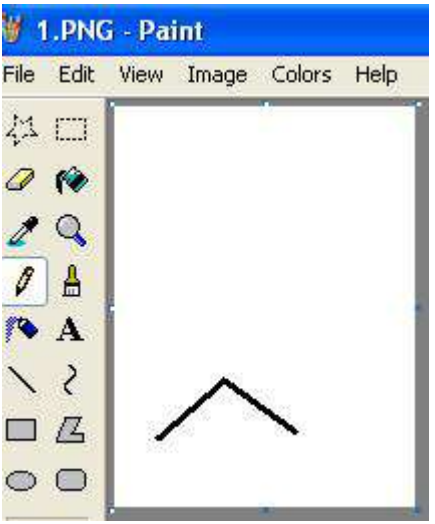
dengan menggunakan utilitas sederhana yang terdapat secara default di dalam Windows yakni Microsoft Paint. Tiap gambar dibuat dengan ukuran yang sama yakni 150 x 200 pixel. Untuk mengganti ukuran di dalam Microsoft Paint, dapat dilakukan dengan mengakses menu *Image* à *Attributes,* lalu mengganti ukuran menjadi yang diinginkan.



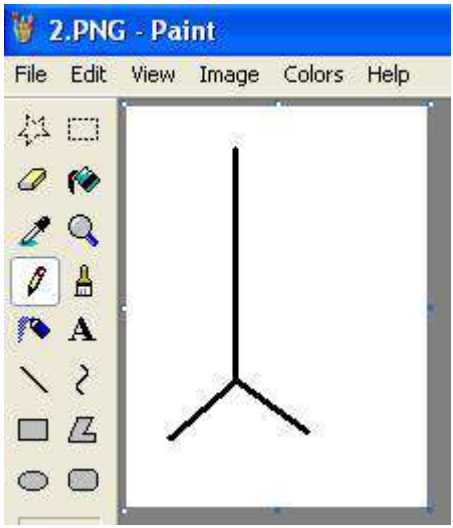


Berikut adalah gambar yang dibuat :

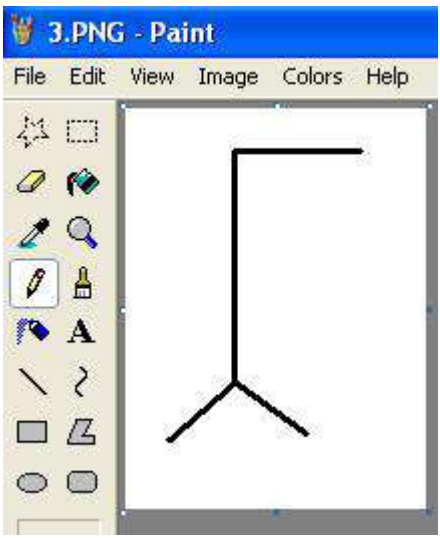
a. 1.png (dengan menggunakan dua buah garis)



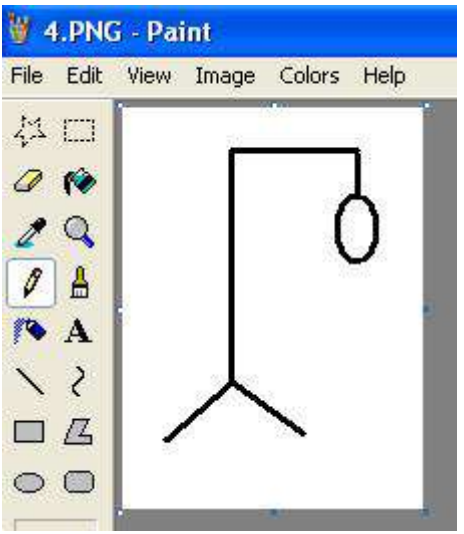
b. 2.png (dapat dilakukan dengan meneruskan gambar yang pertama, kemudian lakukan operasi *Save As*. Cara yang sama juga dilakukan untuk urutan gambar berikutnya demi mendapatkan gambar yang konsisten ukurannya)



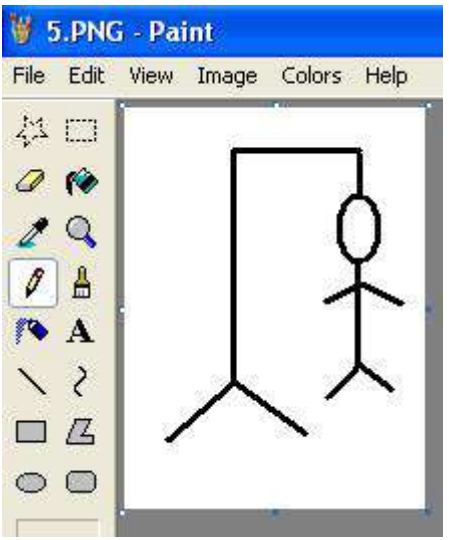
c. 3.png (dengan menambahkan satu garis mendatar lagi)



d. 4.png


4.PNG - Paint
File Edit View Image Colors Help

e. 5.png


5.PNG - Paint
File Edit View Image Colors Help

Anda juga dapat melakukan pembuatan gambar dengan urutan yang berbeda. Urutan tersebut nantinya akan menjadi barometer berapa banyak kesalahan yang ditoleransi dalam permainan. Dengan menempatkan lima gambar, berarti nantinya ada empat toleransi kesalahan dalam satu kali permainan.

2. Kemudian copy seluruh file gambar tersebut ke dalam Solution Explorer. Cara termudah untuk melakukan operasi tersebut adalah dengan melakukan proses *select* dan copy terhadap kelima gambar tersebut dari Windows Explorer dan melakukan operasi *paste* di dalam Solution Explorer. Berikut adalah contoh hasil operasi *copy-paste* kelima file tersebut :

3. Lalu, di dalam form yang tersedia tempatkan komponen-komponen berikut secara berurutan yaitu :

a. Sebuah picturebox dengan setting property *Size* bernilai 150,200
b. Label pertama dengan property *Text* yang isinya dikosongkan dan property *Font* **à** *Size* bernilai 38.

Label pertama atau *Label1* merupakan label yang akan digunakan sebagai pengisian dari tebak kata. Dan nantinya akan berisi karakter *underscore* (_) yang berfungsi sebagai tanda berapa banyak karakter yang harus ditebak oleh pengguna.

c. Label kedua yang isi property *Text*nya adalah *Press letter to guess the word*
d. Label yang ketiga memiliki isi property *Text* **à** *Wrong Letter*.
e. Label yang terakhir atau *Label4* dikosongkan isi property *Text*nya dan memiliki property *Font* **à** *Bold* berisi *true*.
f. Sebuah button dengan property *Text* berisi *Restart*.

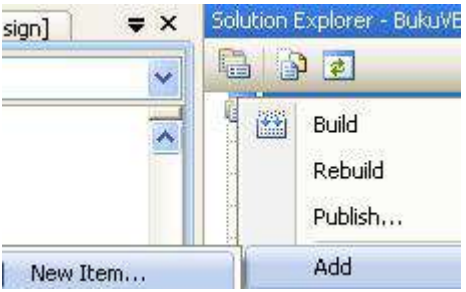
4. Selanjutnya set property dari form dengan nilai berikut :
   a. *MaximizeBox : False*

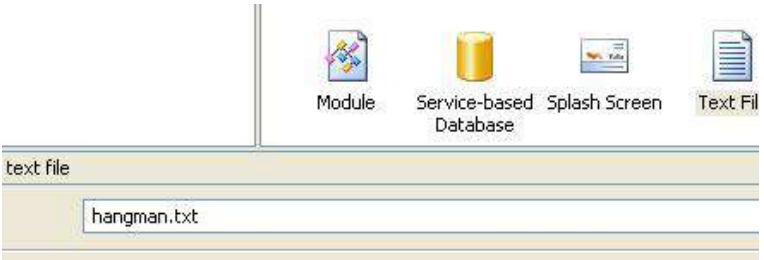
b. *MinimizeBox : False*

c. *KeyPreview : True*

5. Setelah itu tempatkan sebuah komponen imagelist di dalam form tersebut. Set property *ImageSize* menjadi *150, 200*. Pastikan bahwa setting property *ImageSize* dilakukan sebelum penambahan gambar di dalam image list agar ukuran gambar tetap konsisten. Klik tombol kecil pada property *ImageCollection* untuk menambahkan gambar yang sebelumnya sudah dibuat ke dalam property tersebut.

6. Berikutnya, buat sebuah file teks dengan nama *hangman.txt* melalui Solution Explorer. Klik kanan pada root tree dan pilih sub menu *Add* **à** *New Item*. Kemudian pilih jenis *Text File* dan beri nama *hangman.txt*.





7. Di dalam file teks tersebut, ketikkan kata-kata yang akan dijadikan soal untuk game tebak kata. Usahakan agar panjang kata tidak lebih dari tujuh huruf (jika lebih dari tujuh huruf, maka desain form harus diperlebar) dan tiap

kata dipisah dalam baris tersendiri. Buatlah soal dengan jumlah kata yang agak banyak agar terdapat banyak pilihan random pada saat permainan. Berikut adalah contoh kata yang dijadikan soal dalam file *hangman.txt*.

```
VISUAL
BASIC
PROGRAM
CINTA
SAYANG
KASIH
DENDAM
MARAH
EMOSI
```

Berapapun jumlah kata yang dientrikan dalam file teks tersebut tidak akan menjadi masalah dengan asumsi bahwa tiap kata yang akan dijadikan soal dipisah dalam baris tersendiri. Hal ini dikarenakan pada saat pembacaan file teks, akan dipisah dengan menggunakan separator *vbcrlf* atau *Visual Basic Carriage Return Line Feed* yang berarti bahwa tiap kata ditandai dari pemisah baris atau penekanan tombol Enter di saat mengetikkan teks.

8. Set property *Copy to Output* dari file *hangman.txt* menjadi *Copy Always*. Setting ini dilakukan agar file text akan selalu ikut

dicopykan dalam folder yang sama pada saat program dieksekusi.

9. Kini dobel klik pada satu-satunya button yang tersedia dalam form tersebut. Dan ketikkan listing berikut ini :

```
xWord = Split(IO.File. _
    ReadAllText("hangman.txt"), vbCrLf)
Dim acak As New Random
xGuess = xWord(acak.Next(UBound(xWord)))
Label1.Text = ""
For i As Integer = 1 To xGuess.Length
    Label1.Text &= "_ "
Next
ReDim xAda(xGuess.Length - 1)
Array.Clear(xTidak, 0, UBound(xTidak))
Label4.Text = ""
xGagal = 0
PictureBox1.Image = ImageList1.Images(0)
```

Keterangan Listing :

Di dalam listing tersebut, bagian pertama adalah membaca file teks *hangman.txt* yang akan digunakan sebagai soal dalam permainan tebak kata. Selanjutnya isi dari file teks tersebut dipecah ke dalam bentuk array dengan nama *xWord*. Dari kumpulan kata dalam array itu lalu diacak dengan menggunakan sebuah variabel bertipe obyek *random*. Dari kata yang terpilih, kemudian dihitung panjangnya dan ditampilkan di dalam label yang berisi tanda *underscore* (_) sebanyak jumlah huruf yang ada dalam kata tersebut. Di bagian berikutnya dilakukan reset data antara lain : reset data array untuk kegagalan dalam menebak kata (*xTidak*), reset data variabel jumlah kegagalan (*xGagal*) menjadi nilai nol serta deklarasi ulang batas array *xAda*

yang akan berfungsi untuk menghitung jumlah huruf yang sudah ditebak.

10. Kemudian di bagian atas listing setelah baris *Public Class….* Ketikkan deklarasi variabel berikut :

```
Dim xGuess As String
Dim xWord() As String
Dim xAda(), xTidak(4) _
      As String
Dim xGagal As Integer
```

Keterangan Listing :
Di dalam aplikasi ini dibutuhkan beberapa variabel yaitu : variabel *xGuess* untuk menampung huruf yang ditebak oleh pengguna, variabel array *xWord* yang akan merupakan penampung sementara dari soal yang telah dibaca dari file teks, variabel array *xAda* yang digunakan untuk menghitung jumlah huruf yang benar ditebak serta variabel array *xTidak* yang akan diisi oleh huruf yang telah ditebak oleh pengguna dan salah. Sedangkan variabel terakhir adalah variabel *xGagal* yang digunakan untuk menghitung jumlah kegagalan tebakan oleh pengguna.

11. Berikutnya, ketikkan sebuah fungsi untuk membantu dalam melakukan proses pengecekan array. Isi dari fungsi yang bernama *cekArray* tersebut adalah :

```
Function cekArray(ByVal x As Array, _
ByVal s As String) As Boolean
```

```
For j As Integer = 0 To UBound(x)
    If x(j) = UCase(s) _
        Then Return True
Next
End Function
```

Keterangan Listing :
Fungsi ini nantinya digunakan untuk melakukan pengecekan apakah sebuah huruf yang dimaksud ada dalam sebuah array. Sehingga nantinya tidak akan ada huruf dobel yang ditekan oleh pengguna masuk kembali ke dalam permainan.

12. Kini dobel klik pada bagian form yang kosong, sehingga menuju ke prosedur *Form_Load*, lalu ketikkan listing berikut :

```
Button1_Click(sender, e)
```

Keterangan Listing :
Pada saat aplikasi pertama kali dijalankan, maka secara otomatis pengguna akan diarahkan untuk memulai permainan. Agar program berjalan lebih efisien, maka langsung diarahkan agar seakan-akan pengguna telah menekan button *Restart*.

13. Langkah terakhir adalah mengetikkan listing pada prosedur *Form_Keypress*. Untuk menuju ke prosedur tersebut, pilih pada bagian atas dari code explorer dan pilih *Form* pada bagian kiri combobox yang tersedia (class) dan pilih *method KeyPress* di bagian combobox sebelah kanan.

```
Dim xtemp As String
If (e.KeyChar >= Chr(65) And _
    e.KeyChar <= Chr(90)) Or _
    (e.KeyChar >= Chr(97) And _
     e.KeyChar <= Chr(122)) Then
    Dim xOK As Integer = 0
    For i As Integer = 0 To _
        xGuess.Length - 1
        If xAda(i) = "" Then
            If UCase(e.KeyChar) = _
            UCase(Mid(xGuess, i + 1, _
                    1)) Then
                xtemp &= _
                    UCase(e.KeyChar) & " "
                xAda(i) = UCase(e.KeyChar)
                xOK += 1
            Else
                xtemp &= "_ "
            End If
            xOK += IIf(cekArray(xTidak, _
                        e.KeyChar), 1, 0)
            xOK += IIf(cekArray(xAda, _
                        e.KeyChar), 1, 0)
        Else
            xtemp &= xAda(i) & " "
        End If
    Next

    If xOK = 0 Then
        xTidak(xGagal) = _
            UCase(e.KeyChar)
        xGagal += 1
        Label4.Text = ""
        For i As Integer = 0 To _
            UBound(xTidak)
            Label4.Text &= xTidak(i) & " "
        Next
        PictureBox1.Image = _
            ImageList1.Images(xGagal)
    End If
```

```
    Label1.Text = xtemp
    Dim xResult As String
    For i As Integer = 0 To _
        xGuess.Length - 1
        xResult &= xAda(i)
    Next

    If xResult = xGuess Then
        MessageBox.Show _
            ("You Win !!!", "Finish")
        Button1_Click(sender, e)
    End If
    If xGagal >= 4 Then
        MessageBox.Show _
            ("You failed !", _
            "Lose", MessageBoxButtons.OK)
        Button1_Click(sender, e)
    End If
End If
```

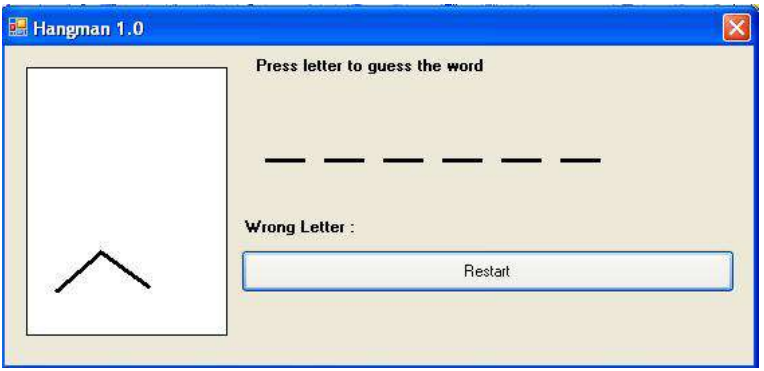
Keterangan Listing :

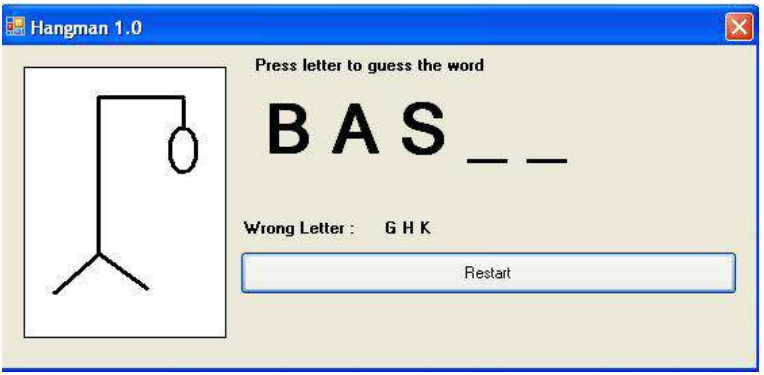
Di dalam prosedur yang menjadi otak dari permainan ini, terbagi menjadi beberapa bagian penting. Bagian pertama adalah melakukan pengecekan apakah tombol keyboard yang ditekan oleh pengguna merupakan alfabet. Pengecekan tersebut dilakukan dengan mengkonversi tombol ke dalam kode ASCII. Anda dapat melihat kode ASCII secara lengkap di dalam MSDN. Hal terpenting yang perlu diketahui untuk kode ASCII dalam listing ini adalah bahwa alfabet terbagi menjadi dua bagian yaitu huruf kecil yang diwakili kode ASCII 65 hingga 90 dan huruf besar yang diwakili kode ASCII 97 hingga 122.

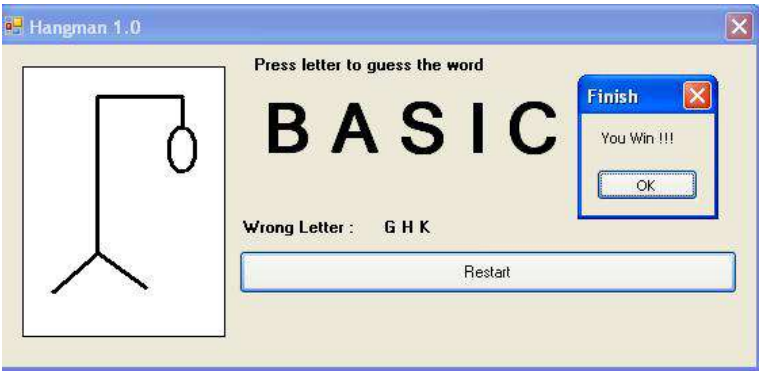
Bagian yang kedua adalah melakukan *parsing* dari karakter yang ditekan oleh pengguna. Jika ternyata karakter tersebut ada di dalam kata yang dijadikan soal, maka akan ditampilkan di dalam label yang menampung kata-kata tersebut. Tetapi jika salah,

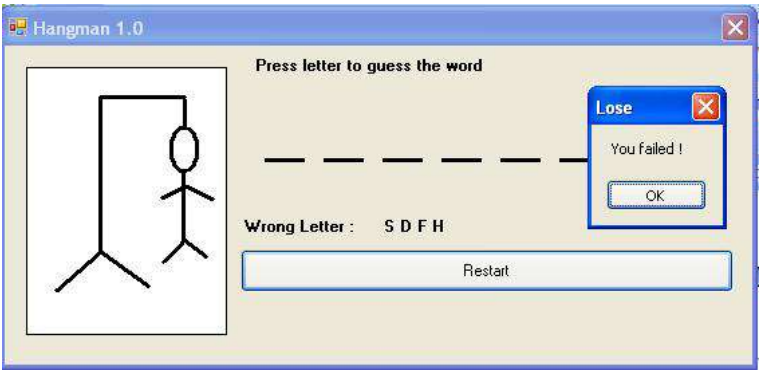
maka akan ditampung di dalam variabel array *xTidak* dan akan dihitung sebagai satu kegagalan di dalam variabel *xGagal*.
Bagian yang ketiga adalah melakukan implementasi jika huruf yang ditekan oleh pengguna ternyata bukan termasuk huruf dalam kata yang ditebak. Maka akan dimasukkan ke dalam variabel penampung sekaligus ditampilkan ke dalam label yang menampilkan huruf yang salah (wrong letter). Selain itu, dilakukan update gambar di dalam komponen picturebox sesuai dengan jumlah kegagalan. Dengan adanya nama gambar yang berurutan (yang telah dibuat sebelumnya), maka secara otomatis pula gambar akan menunjukkan sebuah urutan yang sekuensial.
Bagian terakhir adalah bagian yang melakukan validasi apakah permainan sudah berakhir. Terdapat dua kondisi yakni jika pengguna telah melampaui batas kesalahan sebanyak empat kali maka akan ditampilkan pesan kekalahan. Sedangkan kondisi terakhir adalah kondisi menang jika ternyata semua huruf berhasil ditebak oleh pengguna.

14. Kini eksekusi program tersebut, dan cobalah untuk melakukan tebakan kata berulang kali dengan berbagai kondisi (betul semua, salah semua ataupun betul dan salah).


Hangman 1.0
Press letter to guess the word
Wrong Letter :
Restart


Hangman 1.0
Press letter to guess the word
B A S _ _
Wrong Letter : G H K
Restart


Hangman 1.0
Press letter to guess the word
B A S I C
Finish
You Win !!!
OK
Wrong Letter : G H K
Restart



Saran Pengembangan :
Anda dapat menambahkan fasilitas timer untuk menghitung waktu yang dibutuhkan oleh pengguna dalam menebak kata, sehingga permainan berjalan lebih menarik.

Object Oriented Programming

Contoh 1 : Test Penggunaan Class Sederhana

Level : Menengah
Tujuan :

1. Membuat aplikasi sederhana yang mendemonstrasikan penggunaan *object oriented programming* beserta sifat-sifat yang ada didalamnya.
2. Mengenal penggunaan *encapsulation*
3. Mengenal penggunaan *inheritance*
4. Mengenal penggunaan *polymorphisme*
5. Mengenal cara deklarasi object dari class ke dalam form
6. Mengenal penggunaan enumerasi

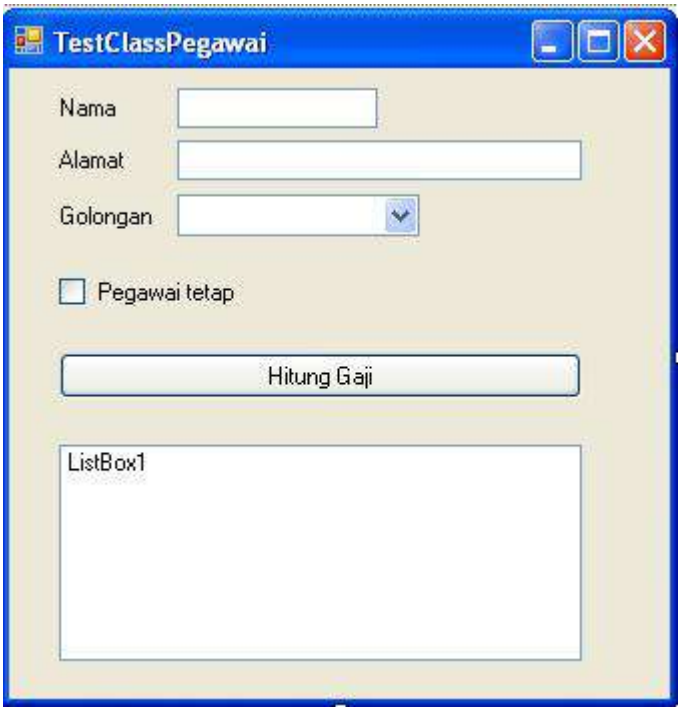
Desain awal :

1. Terdapat tiga buah class yaitu :
    a. Class Pegawai

       Merupakan *base class* yang tidak bisa langsung dideklarasikan sebagai object, sehingga memiliki sifat *MustInherit*
    b. Class PegawaiTetap

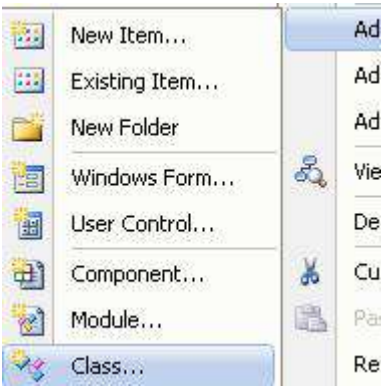
       Merupakan turunan pertama dari class Pegawai, dan memiliki sifat-sifat dasar yang dibutuhkan untuk seorang pegawai.
    c. Class PegawaiKontrak

       Merupakan turunan dari class PegawaiTetap dengan melakukan overload di salah satu fungsi.
2. Terdapat sebuah form yang akan mengimplementasikan penggunaan class tersebut. Desain form :



Langkah awal pembuatan class :

1. Buat sebuah project baru, dan tambahkan sebuah class baru didalamnya dengan melakukan klik kanan pada solution dan memilih sub menu *Add* **à** *Class*. Lalu ketikkan nama *Pegawai* sebagai class pertama yang dibuat.



2. Selanjutnya, di dalam class tersebut, ketikkan listing berikut ini :

```
Public MustInherit Class Pegawai
    Private Nama As String
    Private Alamat As String

    Public Sub setNama(ByVal xNama As String)
        Nama = xNama
    End Sub

    Public Sub setAlamat _
      (ByVal xAlamat As String)
        Alamat = xAlamat
    End Sub

    Public Function getNama() As String
        Return Nama
    End Function


    Public Function getAlamat() As String
        Return Alamat
    End Function
End Class
```

Keterangan listing :
Pada class *Pegawai* sebagai class dasar atau *base class*, maka class tersebut diberikan sifat *MustInherit* yang berarti bahwa class tersebut tidak bisa langsung dijadikan sebagai sebuah object, tetapi harus diturunkan terlebih dulu (diinherit) menjadi class lain. Hal seperti ini biasa dilakukan jika kita ingin mendeklarasikan sebuah class yang memiliki sifat sangat mendasar dan tidak ingin bercampur dengan sifat yang dianggap lebih spesifik.
Di dalam class tersebut juga dideklarasikan beberapa variabel yang bersifat *private* (keterangan lebih lengkap mengenai sifat dari object oriented programming dapat dilihat di appendix) yaitu variabel *nama* dan *alamat* yang dianggap sebagai sifat dasar di dalam class-class turunannya. Karena kedua variabel tersebut bersifat *private*, maka dibuat fungsi yang berguna untuk mengambil nilai serta memberikan nilai kepada kedua variabel tersebut yaitu *setNama* dan *getNama* untuk variabel *nama* dan fungsi *setAlamat* dan *getAlamat* untuk variabel *alamat*.

3. Kemudian, tambahkan lagi class yang kedua yaitu class *PegawaiTetap* dengan menggunakan cara yang sama seperti pada langkah pertama, lalu ketikkan listing berikut ini didalamnya :

```
Public Class PegawaiTetap
    Inherits Pegawai
    Private xtunjangan As Integer
    Private gaji As Integer
    Private tempJabatan As Integer

    Public Enum xJabatan
        Biasa = 0
        Supervisor = 1
```

```
        Manajer = 2
        Direktur = 3
    End Enum

    Public Property Jabatan() As xJabatan
        Get
            Return tempJabatan
        End Get
        Set(ByVal value As xJabatan)
            tempJabatan = value
        End Set
    End Property

    Public Function getJabatan() As String()
        Dim arrayJabatan As Type = _
            GetType(xJabatan)
        Dim tempArray As String
        For Each i As String In _
            [Enum].GetNames(arrayJabatan)
            tempArray &= i & ","
        Next
        tempArray = _
          Left(tempArray, tempArray.Length - 1)
        Return Split(tempArray, ",")
    End Function

    Public Sub setGaji(ByVal xgaji As Integer)
        gaji = xgaji
    End Sub

    Public Function Tunjangan() As Integer
        Select Case tempJabatan
            Case 0
                xtunjangan = 0
            Case 1
                xtunjangan = 250000
            Case 2
                xtunjangan = 750000
            Case 3
                xtunjangan = 1000000
```

```
        End Select
        Return xtunjangan
    End Function

    Public Function Penghasilan() As Integer
        Return gaji + Tunjangan()
    End Function
End Class
```

Keterangan Listing :

Pada class *PegawaiTetap* terdiri dari beberapa bagian yaitu :

a. Deklarasi awal

Pada saat deklarasi awal, terdapat kata kunci *inherits* yang menyatakan bahwa class *PegawaiTetap* merupakan turunan dari class *Pegawai.* Sehingga seluruh sifat yang ada di dalam class *Pegawai* juga dimiliki oleh class *PegawaiTetap*, tetapi class *PegawaiTetap* juga bisa memiliki sifat tambahan tersendiri.

b. Deklarasi variabel dan enumerasi

Di dalam class *PegawaiTetap* terdapat tiga variabel *private* yang berguna untuk membantu deklarasi property serta perhitungan total gaji di dalam fungsi yang akan dijelaskan di bagian berikutnya. Variabel *xtunjangan* digunakan sebagai penampung sementara dalam perhitungan gaji. Sedangkan variabel *gaji* merupakan variabel yang menampung hasil perhitungan gaji, tetapi sengaja dibuat sebagai variabel private untuk menerapkan sifat enkapsulasi. Dan variabel yang terakhir, yaitu variabel *tempJabatan* digunakan untuk membantu implementasi property *Jabatan.* Selain itu terdapat sebuah tipe *enumerasi* (akan dijelaskan lebih lanjut di appendix mengenai OOP). Tipe enumerasi tersebut nantinya akan dimanfaatkan (selain untuk perhitungan gaji, juga untuk mengisi tampilan di form yang menjadi implementasi visual dari class ini). Perlu diingat bahwa apapun isi dari tipe enumerasi, maka nilai yang dikembalikan tetaplah bertipe numerik bulat (integer), sehingga pada

implementasinya (di dalam fungsi *getJabatan* dan perhitungan gaji, percabangan yang dilakukan tetap menggunakan tipe integer). Selain itu, juga perlu diingat bahwa isi dari tipe enumerasi diusahakan tidak mengandung tanda baca didalamnya. Di dalam implementasi visual, property yang memiliki tipe enumerasi nantinya memberikan kebebasan bagi pengguna dalam memilih nilai yang cocok dari sebuah property tanpa harus mengetahui nilai sesungguhnya yang berada di dalam tiap pilihan tersebut.

c. Deklarasi property

Di dalam class *PegawaiTetap* diimplementasikan sebuah property yang nantinya akan berfungsi untuk menempatkan jabatan bagi object pegawai jenis tetap. Property yang memiliki tipe enumerasi ini sesungguhnya mengembalikan nilai bertipe integer dengan default nilai yang dimulai dari angka 0. Implementasi sebuah property umumnya membutuhkan sebuah variabel bantu sebagai penampung sementara sebelum nilai dari property yang diberikan oleh pengguna diolah ulang di dalam class. Pada kasus ini, variabel bantu yang digunakan adalah variabel *xjabatan*.

d. Fungsi untuk mengambil jenis-jenis jabatan

Hasil dari pengisian property jabatan oleh pengguna kemudian diolah di dalam fungsi *Tunjangan* yang memiliki kondisi percabangan untuk tiap jenis jabatan. Dalam kasus ini tiap jenis jabatan diberi permisalan besar tunjangan sendiri-sendiri (hanya sebagai contoh).

e. Fungsi untuk menghitung gaji

Fungsi yang terakhir adalah fungsi sederhana untuk melakukan perhitungan gaji yang terdiri dari dua fungsi yaitu fungsi *setGaji* untuk memberikan nilai gaji pokok dari pegawai dan fungsi *Penghasilan* yang berguna untuk menghitung gaji bersih yang

merupakan hasil penjumlahan dari gaji pokok dengan tunjangan berdasarkan jabatan yang diperoleh.

4. Berikutnya adalah pembuatan class yang terakhir yaitu class *PegawaiKontrak*. Langkah pembuatan class sama dengan langkah pembuatan class sebelumnya di solution yang sama. Isi dari class *PegawaiKontrak* adalah sebagai berikut :

```
Public Class PegawaiKontrak
    Inherits PegawaiTetap
    Public Overloads Function _
      Tunjangan() As Integer
        Return 0
    End Function
End Class
```

Keterangan Listing :
Class *PegawaiKontrak* merupakan class terakhir yang diturunkan dari class *PegawaiTetap* sehingga seluruh sifat yang ada dalam class *PegawaiTetap* diwariskan kedalamnya. Tetapi di dalam class *PegawaiKontrak*, dilakukan modifikasi di fungsi *Tunjangan* dengan menggunakan *overloads* (keterangan lebih lanjut bisa dilihat di appendix). Karena untuk pegawai kontrak memiliki sifat tidak memperoleh tunjangan, maka di class tersebut hasil dari fungsi *Tunjangan* dinolkan.

5. Langkah selanjutnya adalah membuat desain form untuk melakukan testing sifat-sifat class

yang telah dibuat dan mendeklarasikannya ke dalam object-object yang nantinya akan diimplementasikan ke dalam sebuah isian pegawai. Desain form dapat dilihat lagi pada bagian awal contoh ini. Yang pertama kali dilakukan dalam form ini adalah mendeklarasikan dua object dari dua class yang berbeda yaitu object dari class *PegawaiTetap* dan object dari class *PegawaiKontrak* di bagian awal form. Dobel klik pada form, lalu arahkan kursor tepat di bawah deklarasi *Public Class Form …….* dan ketikkan deklarasi berikut ini :

```
Dim xpegawai As New PegawaiTetap
Dim xpegawai1 As New PegawaiKontrak
```

Keterangan Listing :
Deklarasi object dari class sengaja ditempatkan di bagian paling atas agar bisa diakses langsung dari prosedur yang akan dijelaskan di bagian selanjutnya. Perhatikan bahwa deklarasi object selalu diawali dengan kata kunci *New*.

6. Kemudian, di dalam prosedur *Form_Load* ketikkan listing berikut :

```
With ComboBox1
    .Items.AddRange _
        (xpegawai.getJabatan)
    .SelectedIndex = 0
End With
```

Keterangan Listing :
Penambahan item di dalam combobox dilakukan dengan mengambil data yang sudah ada di dalam class *PegawaiTetap*. Fungsi *getJabatan* dari class *PegawaiTetap* mengambil nilai enumerasi *Jabatan*. Selanjutnya indeks dari combobox diarahkan ke item pertama atau item ke-nol.

Isian item dari komponen berjenis *data container* seperti combobox maupun listbox dapat dilakukan satu per satu atau dengan menggunakan sebuah rangkaian data yang dapat dimasukkan ke dalam sebuah array, enumerasi maupun kumpulan object (dengan menggunakan perintah for each…). Jika inputan data dilakukan dengan memanfaatkan rangkaian data maka digunakan method *AddRange*.

7. Selanjutnya buat sebuah prosedur dengan nama *setPegawai* untuk melakukan pengisian data ke dalam masing-masing object. Listing dari prosedur tersebut adalah sebagai berikut :

```
Sub setPegawai _
      (ByVal objectPegawai As Object)
With objectPegawai
    .setNama(TextBox1.Text)
```

```
    .setAlamat(TextBox2.Text)
    .Jabatan = ComboBox1.SelectedIndex
    .setGaji(2500000)
    ListBox1.Items.Clear()
    ListBox1.Items.Add _
        ("Nama : " & .getNama)
    ListBox1.Items.Add _
        ("Alamat : " & .getAlamat)
    ListBox1.Items.Add _
        ("Tunjangan : " & .Tunjangan)
    ListBox1.Items.Add _
        ("Penghasilan : " & .Penghasilan)
End With
End Sub
```

Keterangan Listing :
Langkah yang dilakukan di dalam prosedur tersebut adalah mengakses fungsi-fungsi yang ada di dalam class dan menampilkan hasilnya ke dalam listbox. Inti dari implementasi ini adalah mendemonstrasikan penggunaan sifat *encapsulation* dan memperlihatkan hasilnya ke dalam bentuk visual di sebuah form.

8. Langkah terakhir adalah melakukan dobel klik pada tombol *Hitung Gaji* dan mengetikkan listing berikut didalamnya :

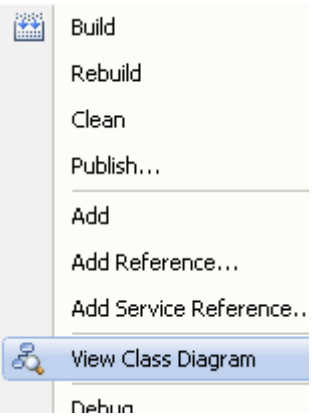
```
setPegawai(IIf(CheckBox1.Checked, _
               xpegawai, xpegawai1))
```

Keterangan Listing :
Pada saat tombol *Hitung Gaji* diklik, maka hal yang pertama dilakukan adalah melakukan pengecekan terhadap checkbox yang menyatakan status pegawai. Jika checkbox tersebut dipilih, maka akan diumpankan object *xpegawai* yang merupakan implementasi dari class *PegawaiTetap*, dan jika tidak dipilih maka akan diumpankan parameter dari *xpegawai1* yang merupakan implementasi dari class *PegawaiKontrak*.

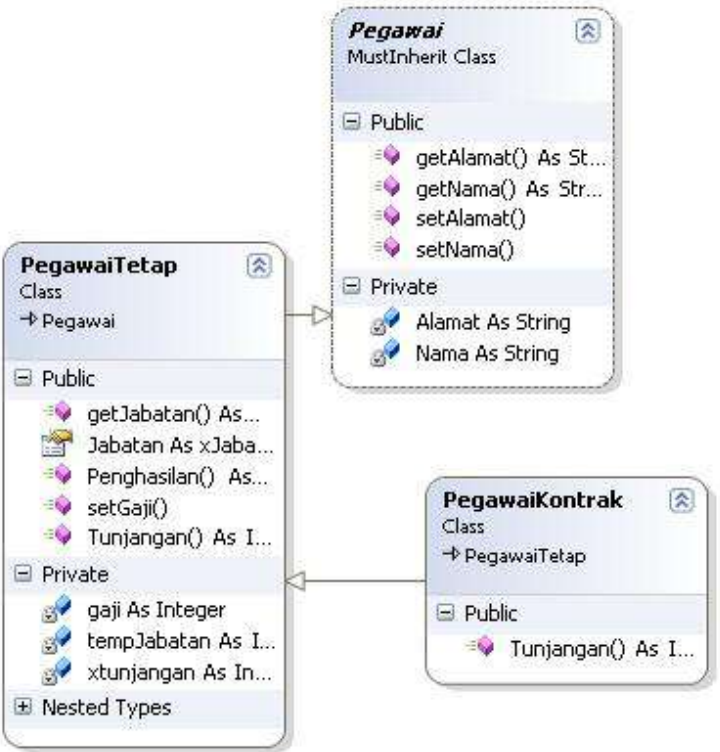
9. Kini, cobalah untuk melakukan testing terhadap form yang telah dibuat. Dan bisa divariasikan beberapa kombinasi, terutama untuk isian di checkbox jenis pegawai (tetap atau kontrak) serta di isian golongan. Sehingga di dalam listbox dapat tertera berbagai jenis jumlah tunjangan serta jumlah total gaji yang diterima oleh karyawan. Salah satu contoh testing tampak pada gambar berikut :

Apabila Anda menggunakan Visual Studio 2008 yang bukan express edition, maka dapat dilakukan proses *reverse engineering* dengan

memilih menu klik kanan dari Solution Explorer dan memilih sub menu *View/Add Class Diagram*.



Maka selanjutnya akan muncul class diagram baru yang didalamnya menyatakan perancangan dari class-class yang telah dibuat, beserta seluruh property dan method yang ada didalamnya dan juga struktur inheritance yang dimilikinya. Dari ketiga class yang sudah dibuat sebelumnya dapat dilihat hasil class diagram yang ada dalam Visual Studio.



Saran Pengembangan :
Anda dapat melakukan perancangan class terlebih dulu sebelum memulai pemrograman. Dengan cara tersebut, maka class yang dihasilkan dapat lebih terorganisir.

Contoh 2 : Tombol Konfirmasi

Level : Menengah
Tujuan :

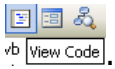
1. Membuat *single user control* yang melakukan inherit dari komponen *button*
2. Mengenal penggunaan *encapsulation*
3. Mengenal penggunaan *inheritance*
4. Mengenal penggunaan *polymorphisme*
5. Mengenal cara implementasi user control ke dalam sebuah form

Desain awal :

1. Sebuah user control dengan inherit dari satu komponen, yaitu button. User control ini nantinya akan berfungsi untuk mengimplementasikan sejenis button *Close* di dalam aplikasi yang akan menampilkan konfirmasi terlebih dulu sebelum aplikasi/form ditutup. Didalamnya terdapat dua jenis pilihan property yang bisa diset, apakah button tersebut akan menutup satu form atau seluruh aplikasi.
2. Hasil dari user control ditempatkan ke dalam sebuah form.
3. User control dan form terletak pada satu solution yang sama.

Langkah penyelesaian :

1. Buat sebuah solution biasa, dan pada Solution Explorer klik kanan dan pilih menu *Add* **à** *User Control*

2. Di dalam kotak dialog yang tersedia, beri nama user control tersebut sebagai *Konfirmasi.vb*.
3. Kemudian, klik icon *View Code* pada Solution Explorer untuk masuk ke dalam mode listing program 

. Dan selanjutnya modifikasi bagian awal dari class user control tersebut seperti pada listing berikut ini :

```
Inherits Button
```

Keterangan Listing :
Dengan melakukan inherits terhadap salah satu komponen (yang umumnya terlihat di dalam Toolbox), maka berarti kita sedang membuat sebuah user control single/simple yang hanya terdiri dari satu turunan komponen (di dalam literatur berbahasa asing umumnya disebut sebagai control). Pada kasus ini, komponen button sebagai class utama yang diturunkan.

4. Lalu, deklarasikan variabel-variabel bantu, termasuk enumerasi yang akan digunakan di dalam property

```
Dim xButuh As Boolean
Dim xPesan As String
Dim xTemp As Integer

Enum xJenis
      FormClose = 0
      FormExit = 1
End Enum
```

Keterangan Listing :
Variabel *xbutuh* digunakan untuk membantu implementasi property *ButuhKonfirmasi* (akan dijelaskan di poin berikutnya). Sedangkan variabel *xpesan* digunakan sebagai alat bantu untuk property *Pesan* yang nantinya akan berisi pesan pada message box konfirmasi (jika property *ButuhKonfirmasi* diberi nilai *true*). Dan variabel *xtemp* digunakan sebagai sebagai variabel bantu di dalam implementasi enumerasi.
Variabel enumerasi yang ada di dalam user control ini sedikit berbeda penggunaannya dibandingkan contoh sebelumnya (class biasa tanpa melibatkan pemrograman visual), karena enumerasi yang digunakan nantinya akan terlihat di dalam jendela property sebagai pilihan di dalam combobox secara langsung.

5. Berikutnya adalah deklarasi property baru dari tombol *Konfirmasi* sebanyak tiga property yaitu property *ButuhKonfirmasi* untuk menandakan apakah saat button diklik akan muncul pesan konfirmasi berupa message box , property *Pesan* yang dapat diisikan pesan jika property *ButuhKonfirmasi* diisi dengan nilai *true*, dan yang terakhir adalah property *AksiButton* yang berfungsi untuk menentukan apakah saat button diklik akan menutup aplikasi atau hanya menutup sebuah form. Listing lengkap dari ketiga property tersebut adalah :

```
Public Property ButuhKonfirmasi() As Boolean
Get
    Return xButuh
End Get
Set(ByVal value As Boolean)
    xButuh = value
End Set
End Property

Public Property Pesan() As String
Get
    Return xPesan
End Get
Set(ByVal value As String)
    xPesan = value
End Set
End Property

Public Property AksiButton() As xJenis
Get
    Return xTemp
End Get
Set(ByVal value As xJenis)
    xTemp = value
End Set
End Property
```

Keterangan Listing :

Dalam contoh ini, memang diterapkan cara yang sangat sederhana dalam melakukan implementasi property di dalam sebuah user control. Dengan cara tersebut memang membutuhkan sebuah variabel bantu (yang telah dijelaskan di poin sebelumnya), tetapi di dalam property jauh terlihat lebih sederhana.

6. Lalu klik pada pilihan event di dalam code editor dan pilih event *KonfirmasiEvents* dan di dalam combobox pilihan untuk method pilih method *click*. Kemudian ketikkan listing berikut ini didalamnya :


```
If xButuh Then
    If MessageBox.Show(xPesan, _
       "Konfirmasi", MessageBoxButtons.YesNo, _
        MessageBoxIcon.Question) = _
        Windows.Forms.DialogResult.Yes Then
        If xTemp = 0 Then
            Parent.FindForm.Close()
        Else
            Application.Exit()
        End If
    End If
End If
```

Keterangan Listing :

Dengan melakukan proses *overriding* pada method *click* (yang merupakan method default pada komponen button, sedangkan untuk komponen yang lain nantinya juga akan disesuaikan dengan method default masing-masing), maka pada saat button *Konfirmasi* diklik (jika sudah berada dalam form) akan mengerjakan isi prosedur yang baru tersebut.

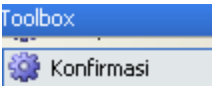
Hal pertama yang dilakukan adalah melakukan pengecekan kondisi percabangan dengan validasi dari isi variabel bantu *xbutuh* yang berasal dari property *ButuhKonfirmasi*, jika memang butuh konfirmasi didalamnya maka akan ditampilkan

messagebox dengan isi pesan yang sesuai dengan isi dari variabel bantu *xpesan* dari property *Pesan*.
Selanjutnya akan dicek isi dari property *AksiButton* melalui isi variabel bantu *xtemp*. Jika berisi dengan nilai nol, maka berarti button hanya akan menutup form, jika berisi nilai satu maka seluruh aplikasi akan ditutup (exit). Cara untuk mengecek form yang sedang aktif atau ditempati oleh button adalah dengan menempatkan kata kunci *Parent* yang berarti induk dari user control saat sudah menjadi object (dalam kasus ini, parent bisa selalu diasumsikan sebagai form).

Satu hal penting yang patut selalu diiingat bahwa dalam pembuatan class non visual, hampir seluruh komponen visual tidak akan dikenali berdasarkan nama yang umumnya muncul pada saat *design time*. Sehingga untuk melakukan akses kepada komponen visual harus menggunakan trik sederhana, yakni dengan mendefinisikan obyek yang memiliki tipe serupa dengan komponen visual tersebut yang nantinya obyek tersebut diakses pada saat ***run time***.

7. Untuk menyelesaikan pembuatan user control pilih menu *Build* **à** *Build [nama solution]*. Proses build ini harus selalu dilakukan jika terjadi perubahan pada user control. Karena user control tidak bisa dilihat efek perubahannya secara langsung, kecuali jika telah ditempatkan di dalam sebuah form.

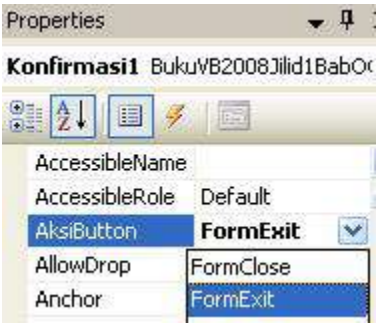
Jika proses build telah berhasil tanpa ada kesalahan, maka beralihlah ke form (yang tersedia secara default di dalam solution). Lalu cek di dalam Toolbox untuk melihat apakah button *Konfirmasi* yang baru dibuat telah muncul di dalam Toolbox tersebut.



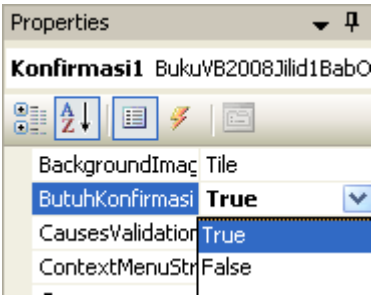
8. Proses berikutnya adalah melakukan testing terhadap user control yang baru saja dibuat. Testing dilakukan dengan melakukan drag and drop dari user control *Konfirmasi* tersebut ke dalam form. Jika button tersebut sudah muncul di dalam form, kini beralihlah ke dalam jendela *Property* di bagian kanan dan perhatikan property-property baru yang telah dibuat sebelumnya.

   Yang pertama adalah property *AksiButton* yang merupakan implementasi dari enumerasi ke dalam sebuah property class (dalam hal ini diwakili oleh user control). Dalam property tersebut akan tampak sebuah combobox

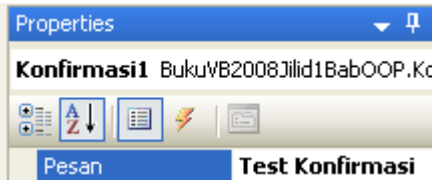
pilihan yang berasal dari enumerasi yang telah dideklarasikan sebelumnya dalam user control.



Kemudian di dalam property *ButuhKonfirmasi* yang bertipe *Boolean*, maka combobox yang tampil hanya berupa pilihan *true* atau *false*.



Dan property baru yang terakhir adalah property *Pesan* yang bertipe *string*, sehingga didalamnya tidak terdapat pilihan, tetapi hanya berupa isian.



Pada saat property yang didefinisikan dalam user control sudah terimplementasi di jendela Property, maka untuk langkah pengembangan selanjutnya dapat dipikirkan property apa yang harus dibuat beserta tipe data apa yang cocok untuk property baru tersebut. Perancangan tersebut seharusnya membutuhkan sebuah teori mengenai perancangan berbasis obyek. Masalah untuk perancangan berbasis obyek dapat dipecahkan dengan menggunakan teori UML (Unified Modeling Language).

9. Setelah seluruh property terisi, maka baru bisa dilakukan proses running aplikasi dengan melakukan penekanan tombol F5. Selanjutnya jika button tersebut diklik akan muncul sebuah messagebox dengan isi kotak pesan sesuai dengan apa yang telah diisikan di property sebelumnya.

   Proses testing dari button buatan sendiri ini sesungguhnya tidak hanya dilakukan dengan menggunakan satu form, sebab untuk

melakukan testing pada property *AksiButton* juga harus dilakukan pada dua buah form. Untuk proses percobaan, diharapkan Anda dapat melakukan perubahan pada property-property baru tersebut dan melakukan proses running berkali-kali untuk memastikan sifat – sifat yang ada di dalam user control tersebut.

Jika Anda menggunakan Visual Studio 2008 non express edition, maka dapat juga dilihat class diagram dari button yang baru saja dibuat dengan menambahkan class diagram ke dalam solution yang sudah dibuat. Class diagram yang tampak adalah sebagai berikut :

Saran Pengembangan :
Anda dapat menambahkan sifat lain atau property baru dari tombol ini seperti model hover.

Contoh 3 : Blinking Label

Level : Menengah
Tujuan :

1. Membuat *composite user control* yang melakukan inherit dari komponen *Timer* dan *Label*
2. Mengenal penggunaan *scope* dalam class
3. Mengenal cara penambahan property dalam composite user control
4. Mengenal cara akses *composite user control* dalam form

Desain awal :

1. Dibuat sebuah user control jenis composite (terdiri dari lebih satu inherit komponen) yaitu dari label dan timer.
2. User control yang dibuat nantinya akan menghasilkan sebuah komponen label baru yang memiliki kemampuan berkedip (berganti warna) tiap satu detik dengan menggunakan bantuan timer.
3. Dari user control jenis composite tersebut, dibuat beberapa property baru agar label yang baru dapat tetap diganti teksnya tetapi tetap mempertahankan sifat enkapsulasi di dalam user control tersebut.
4. Untuk kepentingan testing, dibuat sebuah form yang didalamnya ditempatkan label yang berkedip atau blinking label.

Langkah penyelesaian :

1. Dari sebuah solution biasa, buat sebuah user control baru (lihat langkahnya di contoh sebelumnya) dengan nama *LabelKedip.vb*

2. Kemudian dari template user control yang ada, drag sebuah label dan sebuah timer kedalamnya dan atur sedemikian rupa agar besar dari template user control tampak seperti pada gambar berikut.

Perhatikan bahwa besarnya komponen label di dalam user control tersebut nantinya akan mempengaruhi besar maksimal (*width* dan juga *height*) dari user control saat ditempatkan di dalam form.

3. Lalu klik pada komponen timer dan atur propertynya sebagai berikut :

Enabled	**True**
GenerateMember	True
Interval	**1000**

4. Selanjutnya dobel klik pada template user control untuk masuk ke dalam mode listing. Di dalam user control ini, nantinya terdapat empat buah property baru yang masing-

masing akan mendefinisikan warna pertama dan kedua, teks untuk label serta jenis font untuk label. Sebelum membuat property-property tersebut, terlebih dulu dideklarasikan variabel-variabel bantu untuk keempat property tersebut yang diletakkan tepat di bagian bawah dari deklarasi *Public Class LabelKedip*.

```
Dim xwarna1, xwarna2 As Color
Dim xteks As String
Dim xfont As Font
```

Keterangan Listing :
Deklarasi variabel bantu dalam kasus ini sengaja menggunakan contoh tipe variabel yang memiliki tipe object yaitu *color* dan *font*. Ini untuk memberikan contoh variabel bantu untuk property yang memiliki tipe data balik berupa object. Dan ini juga berarti bahwa sesungguhnya property dalam user control dapat dibuat sefleksibel mungkin sesuai dengan kebutuhan.

5. Kemudian ketikkan listing untuk keempat property yang dibutuhkan sebagai berikut :

```
Public Property Warna1() As Color
Get
    Return xwarna1
End Get
Set(ByVal value As Color)
    xwarna1 = value
End Set
```

```
End Property

Public Property Warna2() As Color
Get
    Return xwarna2
End Get
Set(ByVal value As Color)
    xwarna2 = value
End Set
End Property

Public Property teksLabel() As String
Get
    Return xteks
End Get
Set(ByVal value As String)
    xteks = value
End Set
End Property

Public Property FontLabel() As Font
Get
    Return xfont
End Get
Set(ByVal value As Font)
    xfont = value
End Set
End Property
```

Keterangan Listing :

Di dalam user control untuk label berkedip ini terdapat empat property yaitu :

- Property *Warna1* dan *Warna2* digunakan untuk mengatur warna yang akan digunakan oleh label pada saat berkedip. Karena pada saat berkedip nantinya akan menggunakan dua jenis warna, maka digunakan dua property yang berbeda dengan tipe yang sama yaitu tipe *color*.

- Property *teksLabel* dan property *fontLabel* sesungguhnya memiliki fungsi yang sama dengan property *text* dan *font* di dalam komponen label. Kedua property tersebut diduplikasi karena property yang sebenarnya tidak akan tampak oleh pengguna akibat implementasi sifat enkapsulasi di dalam user control. Jika property dalam label ingin tampak oleh pengguna, maka property *Modifiers* di dalam property label harus diset menjadi *Public*, tetapi hal ini akan menafikan sifat enkapsulasi yang seharusnya menjadi sifat utama dalam pemrograman berorientasi obyek.

6. Selanjutnya, di dalam prosedur *LabelKedip_load* ketikkan listing berikut ini :

```
Label1.Text = xteks
Label1.Font = xfont
```

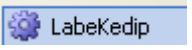
Keterangan Listing :
Saat user control label kedip dijalankan pertama kali, maka akan dibaca terlebih dulu isi dari property *teksLabel* dan *fontLabel* yang dibaca melalui variabel bantu *xteks* dan *xfont*.

7. Dan yang terakhir, dobel klik pada timer dan di dalam prosedur *Timer1_Tick* ketikkan listing untuk pengaturan label kedip.

```
If (Now.Second Mod 2) = 0 Then
    Label1.ForeColor = xwarna1
Else
    Label1.ForeColor = xwarna2
End If
```

Keterangan Listing :
Agar label dapat berkedip tiap detik maka dimanfaatkan komponen timer dengan mengambil sisa hasil bagi dari detik yang berjalan dengan pembagi angka 2. Ini berarti bahwa pada detik ganjil label akan menampilkan warna dari hasil setting property di property *warna2* dan pada detik genap akan menampilkan warna dari hasil setting pada property *warna1*. Hal ini juga dimungkinkan terjadi karena sebelumnya property *interval* di timer telah diset menjadi 1000 dalam satuan milidetik yang berarti sama dengan 1 detik.

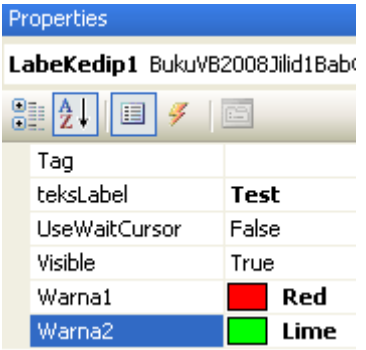
8. User control yang sudah selesai dibuat, tidak bisa langsung digunakan ke dalam form atau dieksekusi (running). Tetapi hanya bisa dilakukan proses *build* agar user control yang sudah selesai dibuat dapat muncul di dalam jendela Toolbox dan bisa didrag ke dalam form.



Umumnya penempatan user control yang baru dibuat terletak di bagian paling atas dari Toolbox. Jika ternyata user control tersebut belum muncul di Toolbox, maka kemungkinan masih terdapat kesalahan di dalam proses *build*, sehingga perlu dicek lagi apakah listing yang diketikkan sudah benar.

9. Kini untuk kepentingan testing, di dalam form yang telah tersedia drag user control tersebut,

cobalah untuk mengubah setting dari property-property yang sudah dibuat yaitu property *fontLabel, teksLabel,Warna1* dan *Warna2*. Serta jangan lupa untuk melakukan proses running demi melihat efek dari user control yang baru dibuat.



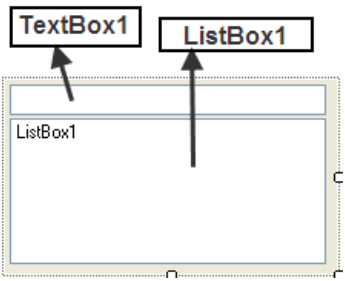
Saran Pengembangan :
Anda dapat menambahkan property yang dapat mengatur level kedip (blinking level) sehingga tidak terpaku pada mode detik.

Contoh 4 : Auto Complete Textbox

Level : Menengah
Tujuan :

1. Membuat *composite user control* yang melakukan inherit dari komponen *TextBox* dan *ListBox*
2. Mengenal penggunaan *scope* dalam class
3. Mengenal cara penambahan property dalam composite user control
4. Mengenal cara akses *composite user control* dalam form

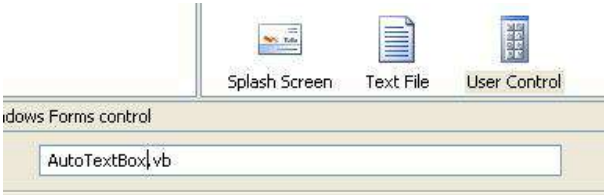
Desain awal :



Langkah Penyelesaian :

1. Buat sebuah solution baru, lalu tambahkan item baru untuk membuat user control dengan melakukan klik kanan pada Solution Explorer dan memilih sub menu *Add* **à** *New Item*.

2. Dalam kotak dialog yang tersedia, pilih tipe *User Control* dan beri nama *AutoTextBox*.



Perlu diingat sekali lagi, seperti halnya pada contoh sebelumnya, maka user control yang dibuat kali ini adalah jenis *composite user control* yang berarti di

dalam user control tersebut nantinya akan berisi lebih dari satu hasil *inherit* komponen visual yang sudah tersedia secara ***built in*** dari .NET Framework.

3. Berikutnya di dalam template yang sudah tersedia di user control tersebut, drag sebuah textbox dan listbox dengan susunan seperti pada gambar.
4. Set property *Visible* dari listbox yang ada dengan nilai *false* dan pastikan bahwa lebar listbox dan textbox sama persis demi memberikan efek *autocomplete*.

Efek ***autocomplete*** dapat dilihat pada saat Anda melakukan proses programming di Visual Studio. Juga seringkali kita lihat pada situs-situs modern seperti halnya Google maupun Yahoo yang seakan bisa menebak apa yang akan diketikkan oleh pengguna. Dalam contoh ini, teks yang dijadikan acuan efek autocomplete dibaca dari file teks. Tetapi dalam pengembangannya, bisa ditempatkan data yang berasal dari database untuk menjadi autocomplete menjadi lebih bervariasi dan sesuai kebutuhan.

5. Kemudian di dalam bagian pengetikan listing atau *code view*, ketikkan listing berikut di

bagian paling atas listing, tepat di bawah deklarasi *Public Class*….

```
Dim xTeks As String
Public Property FileTeks() _
       As String
Get
    Return xTeks
End Get
Set(ByVal value As String)
    xTeks = value
End Set
End Property
```

Keterangan Listing :
Pembuatan property *FileTeks* nantinya akan diisi dengan nama file teks yang akan dijadikan acuan dalam proses pemberian efek autocomplete. Sedangkan variabel *xTeks* merupakan variabel bantu yang digunakan sebagai penampung sementara dari isi nilai property *FileTeks.*

Di dalam pengembangan user control ini, dapat juga diberikan nilai acuan dari sumber yang variatif. Sebagai contoh, dapat diberikan sumber data dari array, field sebuah tabel maupun dari cookies browser.

6. Berikutnya dobel klik pada satu-satunya textbox yang tersedia sehingga menuju ke prosedur *TextBox1_TextChanged* lalu ketikkan listing berikut :

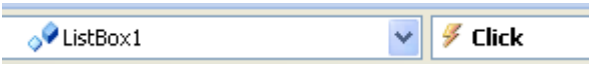
```
If TextBox1.Text <> "" Then
    If IO.File.Exists(xTeks) Then
        Dim xtemp() As String = _
            Split(IO.File. _
                  ReadAllText(xTeks), vbCrLf)
        Dim xComplete As String = _
            TextBox1.Text
        Dim xAda As Boolean = False
        ListBox1.Items.Clear()
        For i As Integer = 0 To _
            UBound(xtemp) - 1
            If UCase(xComplete) = _
                UCase(Mid(xtemp(i), 1, _
                    xComplete.Length)) Then
                ListBox1.Visible = True
                ListBox1.Items.Add(xtemp(i))
                xAda = True
            End If
        Next
        If Not xAda Then _
            ListBox1.Visible = False
    Else
        ListBox1.Items.Clear()
        ListBox1.Visible = False
    End If
End If
```

Keterangan Listing :

Inti dari listing tersebut adalah melakukan deteksi terhadap apa yang telah diketikkan oleh pengguna di dalam textbox. Jika ternyata terdapat sebuah karakter yang diketikkan, maka selanjutnya akan dibaca file teks yang sudah didefinisikan namanya di dalam property *FileTeks*. Lalu dilakukan proses validasi dari tiap item yang terdapat dalam file teks tersebut, jika memang terdapat kecocokan, maka listbox akan ditampilkan beserta daftar item yang dianggap sesuai. Tetapi jika tidak, maka listbox akan disembunyikan atau dalam mode *hidden.*

Untuk pengembangan, dapat pula ditambahkan fitur *cookies autocomplete* atau melakukan penyimpanan isian terakhir yang diketikkan oleh pengguna. Sehingga akan terjadi efek autocomplete yang seakan-akan mampu mengingat kebiasaan pengguna di dalam sebuah aplikasi. Penyimpanan tersebut dapat diarahkanke sebuah file konfigurasi ataupun file teks biasa.

7. Berikutnya, dobel klik pada listbox, kemudian arahkan ke prosedur *Listbox_Click* dengan mengganti method di bagian atas code view, lalu ketikkan listing berikutnya.




```
If ListBox1.Items.Count > 0 Then
    If ListBox1.SelectedIndex _
        >= 0 Then
        TextBox1.Text = _
            ListBox1.Text
        ListBox1.Visible = False
    End If
End If
```

Keterangan Listing :
Dengan memanfaatkan prosedur *Listbox_Click* berarti bahwa pada saat listbox diklik oleh pengguna maka isi item dari listbox yang diklik akan dicopy ke dalam textbox.

8. Berikutnya, buatlah sebuah file teks untuk kepentingan testing komponen yang baru saja dibuat. File teks yang dijadikan contoh di sini adalah sebuah rangkaian data nama dan file teks tersebut diberi nama *NamaKaryawan.txt*. Pembuatan file teks dilakukan di dalam lingkup Visual Studio dengan melakukan klik kanan di Solution Explorer dan memilih sub menu *Add* **à** *New Item* dan selanjutnya memilih item *Text File*. Isi dari contoh file teks tersebut adalah sebagai berikut (juga dapat dimodifikasi sesuai selera) :

```
Ali
Adi
Ari
Budi
Benu
Bustomi
Caca
Cici
Didi
Dedi
Deni
Eka
Erna
Eli
```

9. Langkah berikutnya adalah melakukan setting property *Copy Output* dari file teks yang baru

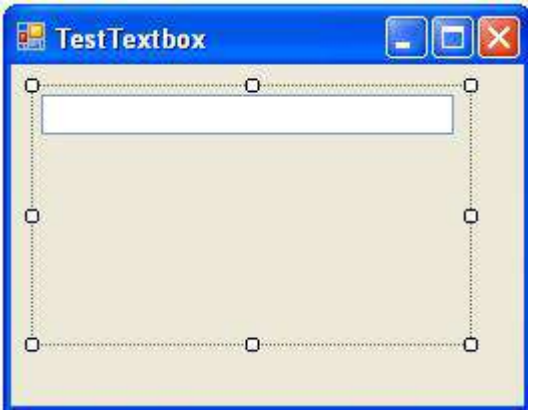
saja dibuat menjadi *Copy Always*. Hal ini untuk memastikan bahwa nantinya file teks tersebut akan berada di dalam folder yang sama dengan hasil eksekusi aplikasi.

10. Kini saatnya untuk melakukan proses *Build* agar user control yang sudah selesai dibuat dapat muncul ke dalam Toolbox. Lakukan proses tersebut dengan memilih menu *Build* **à** *Build [nama project]*. Setelah selesai, dapat dicek di dalam Toolbox, apakah user control yang dimaksud telah masuk didalamnya.

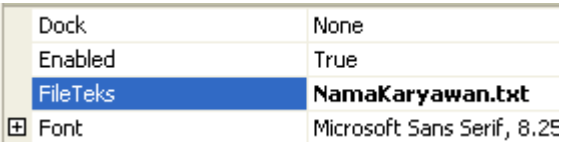
Sekali lagi, sangat penting diperhatikan apakah user control telah berhasil masuk ke dalam Toolbox. Jika memang belum berhasil, perlu dicek ulang langkah-langkah pembuatan user control sebelumnya.

11. Dan berikutnya, untuk kepentingan testing, di dalam form yang tersedia drag user control tersebut ke dalam form. Perhatikan bahwa besarnya user control sedikit lebih besar karena ukuran listbox juga dimasukkan meski

secara kasat mata masih belum terlihat di saat *design time*.



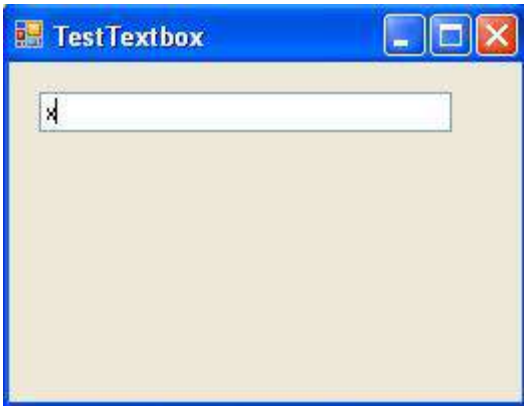
12. Di dalam user control yang didrag tersebut, set nilai property *FileTeks* menjadi *NamaKaryawan.txt*.



Untuk kepentingan testing lebih lanjut, Anda dapat melakukan drag dua user control yang menggunakan file teks yang berbeda-beda. Hal ini demi memastikan apakah user control tersebut benar-benar valid.

13. Setelah selesai, kini saatnya untuk menjalankan aplikasi tersebut. Anda dapat

berusaha mencoba mengisikan berbagai variasi nama untuk mengetahui apakah textbox yang sudah dibuat telah berfungsi atau belum. Jika diketikkan sebuah karakter yang terdapat di dalam rangkaian data nama, maka listbox akan muncul, sedangkan jika tidak ada karakter di dalam rangkaian data yang dimaksud maka listbox akan tetap dalam mode hidden.



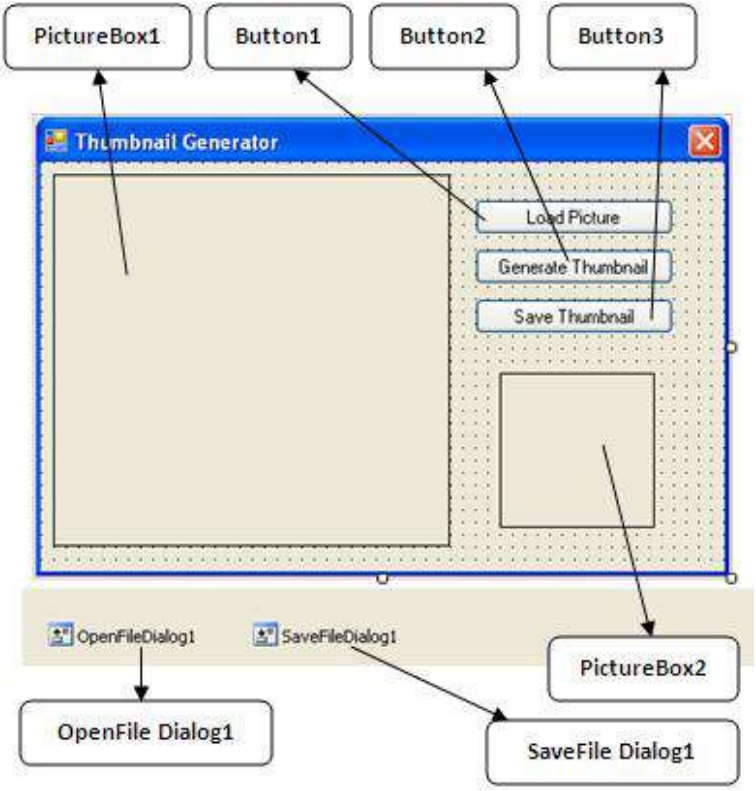
Saran Pengembangan :
Anda dapat mengarahkan proses pengisian data dari listbox ke textbox dengan berbagai jenis method seperti dengan menggunakan tombol Enter ataupun dengan menekan tombol spasi seperti layaknya fasilitas autocomplete di dalam Visual Studio.

Aplikasi Pengembangan

Contoh 1 : Thumbnail Generator

Level : Mahir
Tujuan :
1. Mengenal penggunaan GDI+
2. Menggunakan PictureBox sebagai penampung hasil manipulasi GDI+

Desain awal :



Langkah Penyelesaian :

1. Buat sebuah solution baru dengan nama *thumbnailGenerator*. Di dalam form yang telah tersedia tersebut, tempatkan komponen-komponen berikut ini seperti pada gambar desain awal :

a. Dua buah komponen picturebox
b. Tiga buah button
c. Sebuah komponen openfiledialog dan sebuah komponen savefiledialog

2. Pada picturebox yang pertama set property *Size* **à** *Width* menjadi 250 dan *Size* **à** *Height* menjadi 240. Lalu set property *BorderStyle* menjadi *FixedSingle* dan property *SizeMode* ke nilai *StretchImage*.

Anda dapat mengganti nilai property *Size* sesuai selera yang secara otomatis akan berpengaruh juga ke ukuran form yang dibuat. Inti dari pengaturan property ini adalah selera dari ukuran proporsional yang dikehendaki pada saat perancangan.

3. Picturebox yang kedua memiliki property yang mirip dengan picturebox pertama, yakni property *BorderStyle* menjadi *FixedSingle* dan property *SizeMode* ke nilai *StretchImage*. Tetapi untuk property *Size* **à** *Width* 250 dan *Size* **à** *Height* diset menjadi 100.
4. Di komponen openfiledialog set property *DefaultExt* menjadi **.jpg*|**.bmp*.

5. Kini dobel klik di button yang pertama dan berfungsi untuk menampilkan gambar yang akan dijadikan thumbnail. Thumbnail adalah gambar tiruan dalam versi yang lebih kecil dan ringan saat terjadi proses loading. Umumnya digunakan dalam sebuah aplikasi ataupun situs yang membutuhkan proses preview gambar yang berukuran besar. Dalam button yang pertama tersebut ketikkan listing berikut ini :

```
Dim xFile As String = ""
Dim xValid As Boolean = True
With OpenFileDialog1
    If .ShowDialog = _
        DialogResult.OK _
        And .FileName <> "" Then
        xFile = .FileName
    Else
        xValid = False
    End If
End With

If xValid Then
    If xGambar IsNot _
        Nothing Then _
        xGambar.Dispose()
    xGambar = New Bitmap(xFile)
    PictureBox1.Image = _
        CType(xGambar, Image)
End If
```

Keterangan Listing :
Di dalam listing tersebut, terbagi menjadi dua bagian yaitu membuka file gambar dan mengarahkan file gambar yang terbuka ke obyek dan picturebox. Di bagian yang pertama, dengan memanfaatkan komponen openfiledialog, maka file gambar akan dibuka dengan batasan hanya format JPEG dan BMP. Sedangkan di bagian yang kedua, file gambar yang telah dibuka akan diarahkan ke obyek tipe bitmap dan ditampilkan ke dalam picturebox.

Dalam pengembangannya juga dapat dibuka file gambar tipe lain seperti .PNG maupun .GIF. Di pengembangannya, juga dapat ditambahkan fasilitas untuk melakukan validasi, apakah file gambar yang dibuka memang layak untuk dijadikan thumbnail. Karena jika file gambar yang dibuka terlalu kecil atau lebih kecil dari ukuran thumbnail, maka proses pembuatan thumbnail akan terlihat sia-sia.

6. Selanjutnya ketikkan deklarasi variabel di bagian bawah setelah deklarasi *Public Class….*

```
Dim xGambar, xThumb As Image
```

Keterangan Listing :
Dua variabel yang dideklarasikan keduanya bertipe *Image*. Variabel yang pertama (*xGambar*) nantinya akan diisi dengan gambar yang asli, sedangkan variabel yang kedua (*xThumb*) akan diisi dengan hasil sementara proses pembuatan thumbnail.

7. Lalu dobel klik pada button yang kedua yang merupakan inti dari aplikasi *thumbnail generator* ini, dan ketikkan listing berikut :

```
If xGambar IsNot Nothing Then
    xThumb = xGambar.GetThumbnailImage _
        (100, 100, Nothing, _
        New IntPtr())
    PictureBox2.Image = _
        CType(xThumb, Image)
End If
```

Keterangan Listing :
Dalam proses inti pembuatan thumbnail, variabel *xThumb* diisi dengan sebuah method bernama *GetThumbnailImage* yang akan secara otomatis membuat sebuah file thumbnail dengan ukuran yang telah ditentukan. Dalam contoh listing tersebut, ukuran thumbnail didefinisikan dengan ukuran 100 x 100 pixel.

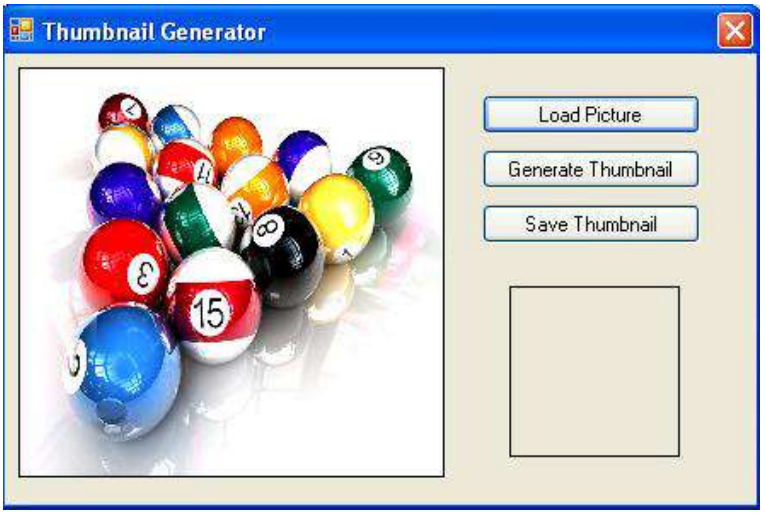
Ukuran file thumbnail yang paling ideal adalah dengan menghitung secara proporsional dari ukuran gambar yang akan dijadikan thumbnail. Anda bisa menggunakan rumus sederhana dengan melakukan perkalian agar lebar dan tinggi dari gambar dapat secara proporsional mengecil.

8. Yang terakhir adalah melakukan dobel klik pada button yang berguna untuk menyimpan gambar hasil proses dari button yang kedua. Listing yang terdapat dalam button ini adalah :

```
If xThumb IsNot Nothing Then
    With SaveFileDialog1
        If .ShowDialog = DialogResult.OK _
            And .FileName <> "" Then
            xThumb.Save(.FileName & ".gif", _
                Imaging.ImageFormat.Gif)
        End If
    End With
End If
```

Keterangan Listing :
Listing yang terakhir melakukan fungsi sederhana yakni menyimpan hasil proses pembuatan gambar thumbnail ke dalam sebuah file gambar lain dengan format GIF. Pemilihan format GIF dikarenakan format ini merupakan format gambar dengan ukuran terkecil. Dan meski memiliki kelemahan di dalam hal penyimpanan kuantitas dan kualitas warna, tetapi untuk sebuah gambar yang berukuran kecil, maka pemilihan format tersebut adalah yang terbaik.

9. Untuk melakukan proses testing dari aplikasi sederhana ini, dapat dilakukan validasi proses kepada beberapa file gambar dengan ukuran yang bervariasi. Contoh tampilan dari proses pembuatan file thumbnail tampak pada gambar berikut :



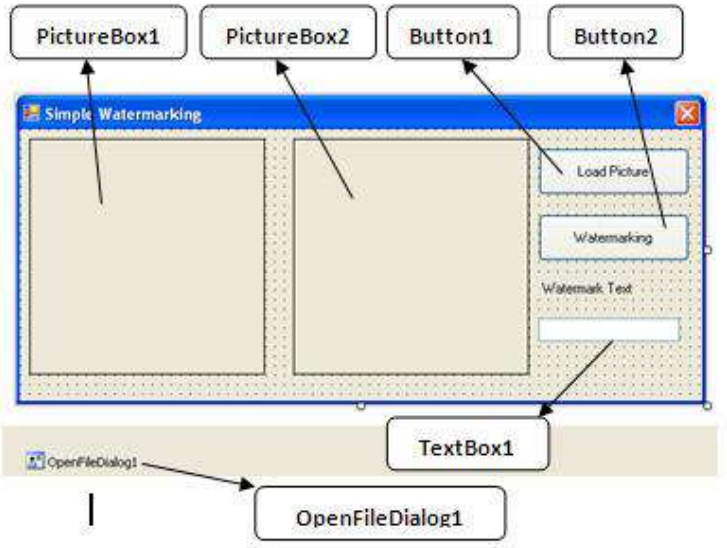
Saran Pengembangan :
Dari aplikasi tersebut, dapat dikembangkan dengan membuat sebuah proses pembuatan gambar thumbnail secara *batch* atau

borongan. Dan dari hasil proses batch tersebut dapat dimanfaatkan sebagai sebuah aplikasi untuk membuat album preview dari sebuah folder yang memuat foto dengan kuantitas sangat banyak.

Contoh 2 : Watermarking Image

Level : Mahir
Tujuan :
1. Memanfaatkan GDI+ dalam melakukan manipulasi sebuah file gambar
2. Memanfaatkan manipulasi teks sebagai sebuah obyek graphic dan menggabungkan dengan sebuah file gambar.

Desain awal :



Langkah penyelesaian :

1. Buat sebuah solution baru dan di dalam form yang telah tersedia tempatkan komponen-komponen berikut sesuai dengan desain pada gambar :
    a. Dua buah picturebox yang keduanya memiliki setting property sama yaitu :
        i. *BorderStyle* **à** *FixedSingle*
        ii. *Size* **à** *200,200*
        iii. *SizeMode* **à** *Stretch Image*

b. Dua buah button dengan property *Text* masing-masing berisi nilai *Load Picture* dan *Watermarking*.
c. Sebuah label dengan property *Text* berisi *Watermark Text*.
d. Sebuah textbox dengan property *Text* yang dikosongkan.
e. Yang terakhir adalah sebuah komponen openfiledialog.

2. Langkah berikutnya adalah melakukan dobel klik di button yang pertama atau button *Load Picture*, lalu ketikkan listing berikut ini :

```
Dim xFile As String = ""
Dim xValid As Boolean = True
With OpenFileDialog1
    If .ShowDialog = DialogResult.OK _
        And .FileName <> "" Then
        xFile = .FileName
    Else
        xValid = False
    End If
End With
If xValid Then
    ImageDispose(xGambar)
    xGambar = New Bitmap(xFile)
    PictureBox1.Image = _
        CType(xGambar, Image)
End If
```

Keterangan Listing :
Mirip dengan contoh sebelumnya (thumbnail generator), langkah pertama adalah melakukan proses loading file gambar dengan memanfaatkan komponen openfiledialog yang kemudian dimasukkan ke dalam obyek bertipe image sekaligus menampilkan ke picturebox yang pertama. Hal yang berbeda adalah pada saat proses validasi apakah obyek image telah terisi atau belum oleh file gambar, ditempatkan di sebuah prosedur dengan nama *ImageDispose*. Isi dari prosedur tersebut dapat dilihat pada langkah selanjutnya.

Di dalam proses validasi pemanggilan file gambar, dapat dikembangkan dengan mempertimbangkan beberapa faktor seperti apakah benar bahwa file gambar yang akan diberikan watermark memang cukup besar ukurannya, dan juga dapat ditambahkan validasi untuk format file gambar yang dianggap cukup layak untuk diberikan watermark.

3. Kemudian deklarasikan variabel berikut di bagian atas listing tepat di bawah deklarasi *Public Class….*

```
Dim xGambar, xWater As Image
```

Keterangan Listing :
Kedua variabel bertipe obyek image tersebut nantinya akan digunakan sebagai penampung file gambar yang asli (*xGambar*) dan gambar yang telah disisipi teks didalamnya (*xWater*).

4. Berikutnya ketikkan listing dari prosedur *ImageDispose* yang berguna untuk *membuang* obyek bertipe image dari parameter pemanggil. Proses pembuangan atau *dispose* tersebut mencegah pemakaian memori yang berlebihan dari sebuah obyek, terutama jika sebelumnya telah dilakukan proses loading file gambar yang berukuran besar.

```
Sub ImageDispose(ByVal x As Image)
      If x IsNot Nothing Then _
          x.Dispose()
End Sub
```

5. Kini untuk listing yang terakhir, dobel klik pada button *Watermarking* untuk mengetikkan listing berikut ini :

```
If xGambar IsNot Nothing Then
    Dim xTulisan As String
    If TextBox1.Text <> "" Then
        ImageDispose(xWater)
        xWater = New Bitmap(xGambar)
        Dim xTempFile As String = _
            "c:\temp.jpg"
        Dim g As Graphics = _
             Graphics.FromImage(xWater)
        Using xWater
            Dim xfontsize As Single = _
                (xWater.Height * (20 / 100))
            Dim font As New Font _
```

```
                    ("verdana", xfontsize, _
                    FontStyle.Bold)
            Dim xsize As  _
                System.Drawing.SizeF = _
                g.MeasureString _
                (TextBox1.Text, font)
            Dim z As New  _
                System.Drawing. _
                Drawing2D.Matrix
            z.Reset()
            g.DrawString(TextBox1.Text, _
            font, Brushes.Black, 10, _
                xWater.Height / 2)
            xWater.Save(xTempFile)
            ImageDispose(xWater)
            xWater = New Bitmap(xTempFile)
            PictureBox2.Image = _
                CType(xWater, Image)
        End Using
    End If
End If
```

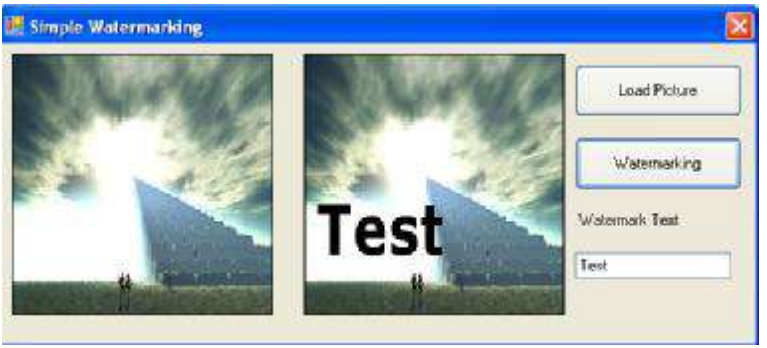
Keterangan Listing :

Di dalam listing tersebut terdapat dua bagian utama dalam proses yang dilakukan yakni : membuat watermark dari teks yang telah dientrikan di dalam textbox dan menggabungkannya ke dalam sebuah file gambar temporer. Pembuatan teks dengan mode grafis tersebut dengan melakukan perhitungan ukuran tinggi dari file gambar yang kemudian 20% dari ukuran tersebut dijadikan acuan untuk besarnya font dari teks yang akan ditempatkan. Selanjutnya, teks tersebut ditempatkan dalam gambar yang sama di obyek image (*xWater*) sehingga nantinya akan menjadi watermark dari gambar tersebut. File gambar yang akan diberikan watermark terlebih dulu disimpan secara temporer ke dalam sebuah file di drive c: dengan nama *temp.jpg*.

Dan langkah yang terakhir adalah menampilkan hasil proses watermark ke dalam picturebox yang telah tersedia.

Terdapat beberapa saran pengembangan yang bisa dilakukan di dalam proses ini, seperti membuat opsi penempatan teks watermark. Anda dapat membuat teks watermark memiliki posisi di atas, bawah maupun berulang secara diagonal. Pengulangan teks watermark akan mencegah seseorang yang berusaha menghilangkan watermark tersebut dengan aplikasi pengolah gambar. Pengembangan lainnya dapat dilakukan dengan membuat efek teks menjadi efek yang tergolong *non solid*. Hal tersebut dapat dilakukan dengan membuat efek shadow maupun emboss terhadap teks yang akan ditempatkan di dalam file gambar.

6. Berikut ini adalah contoh *screen capture* dari aplikasi untuk menambahkan watermark teks pada file gambar.



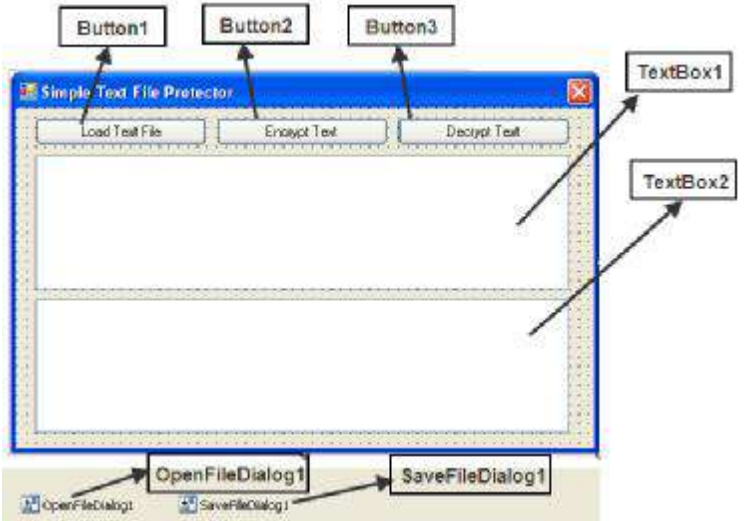
Saran Pengembangan :

Aplikasi ini juga dapat dikembangkan dengan menambahkan kemampuan untuk proses *batch watermarking* atau melakukan proses watermarking secara borongan terhadap banyak file gambar sekaligus.

Contoh 3 : Text File Protector

Level : Mahir
Tujuan :
1. Mengenal penggunaan enkripsi dengan *built in class* dari .NET Framework
2. Mengenal penyimpanan key dan IV dalam algoritma enkripsi agar dapat didekrip ulang

Desain awal :



Langkah penyelesaian :

1. Buat sebuah solution baru, dan di form yang tersedia, tempatkan komponen visual sesuai dengan tata letak yang tertera pada gambar :
    a. Tiga buah button yang masing-masing property *Text*-nya diisikan dengan nilai *Load File Text*, *Encrypt Text* dan *Decrypt Text*.
    b. Dua buah textbox yang keduanya memiliki setting property sama yakni :
        i. *Multiline* **à** *True*

ii. *ScrollBars* **à** *Vertical*

c. Sebuah komponen openfiledialog dengan setting property *FileName* yang isinya dikosongkan, serta property *Filter* yang diisi dengan nilai *Text Files (*.txt)|*.txt*

d. Sebuah komponen savefiledialog dengan nilai property *Filter* yang sama dengan isi property *Filter* dari komponen openfiledialog.

2. Langkah pertama adalah menuju ke code view dan di bagian paling atas, ketikkan listing berikut sebagai bantuan sebelum mengetikkan listing berikutnya :

```
Imports System.Security.Cryptography
Imports System.Text
```

Keterangan Listing :
Kedua referensi yang diketikkan pertama kali tersebut untuk membantu agar saat pengetikan tidak terlalu panjang sekaligus sebagai petunjuk bahwa nantinya di listing berikutnya akan menggunakan sintaks dan *built in* class dari kedua library tersebut.

Perlu diingat bahwa aplikasi ini nantinya akan melakukan proses enkripsi dari sebuah file teks dengan algoritma Triple DES. Salah satu perbedaan utama dari

contoh ini dibandingkan kebanyakan contoh aplikasi serupa yang tersebar di internet adalah bahwa aplikasi ini juga menyertakan key dan *IV* dari hasil enkripsi di dalam file yang dienkripsi. Sehingga proses dekripsi dapat dijalankan dalam waktu yang berbeda tetapi tetap akan mendapatkan hasil yang akurat. Penyimpanan tersebut dilakukan karena umumnya key dan IV hanya didapat dari hasil generate secara langsung dari fasilitas yang tersedia, sehingga proses enkrip-dekrip hanya dapat dijalankan dalam satu satuan waktu.

3. Kemudian dobel klik pada button *Load Text File* dan ketikkan listing berikut didalamnya.

```
With OpenFileDialog1
    If .ShowDialog() = DialogResult.OK _
        And .FileName <> "" Then
        TextBox1.Text = IO.File. _
            ReadAllText(.FileName)
    End If
End With
```

Keterangan Listing :
Proses yang ada dalam listing ini sangat sederhana yakni hanya melakukan proses loading file dengan format teks (.txt) ke dalam textbox yang tersedia. Selanjut file teks tersebut yang nanitnya akan dikenai proses enkripsi ataupun proses dekripsi, tergantung dari langkah selanjutnya yang dilakukan oleh pengguna.

4. Berikutnya ketikkan deklarasi variabel berikut tepat di bawah deklarasi *Public Class…*

```
Dim xkey, xiv As Byte()
```

Keterangan Listing :
Kedua variabel tersebut nantinya akan digunakan sebagai penampung sementara dari key dan IV yang akan digenerate pada saat proses enkripsi. Sedangkan di dalam proses dekripsi, key dan IV akan dibaca dari dua baris terakhir file teks (jika file teks tersebut memang benar-benar dalam keadaan terenkripsi). Key dan IV yang sejatinya memiliki format byte nantinya akan dikonversi ke dalam bentuk string format *base-64* agar dapat dimasukkan ke dalam file teks (untuk proses enkripsi) dan juga dibaca (untuk proses dekripsi).

5. Lalu dobel klik pada button yang kedua atau button untuk melakukan proses enkripsi. Di dalam prosedur *Button1_Click* ketikkan listing berikut ini :

```
If TextBox1.Text <> "" Then
    TextBox2.Clear()
    Dim ct As ICryptoTransform
    Dim ms As IO.MemoryStream
    Dim cs As CryptoStream
    Dim byt() As Byte
    Dim mcsp As SymmetricAlgorithm
    Dim keystring, ivstring As String

    Dim xvalue As String = TextBox1.Text
    mcsp = New RC2CryptoServiceProvider
    mcsp.GenerateKey()
```

```
    xkey = mcsp.Key
    keystring = Convert.ToBase64String _
          (mcsp.Key)
    mcsp.GenerateIV()
    xiv = mcsp.IV
    ivstring = Convert.ToBase64String(mcsp.IV)
    xkey = Convert.FromBase64String(keystring)
    xiv = Convert.FromBase64String(ivstring)
    ct = mcsp.CreateEncryptor _
       (mcsp.Key, mcsp.IV)
    byt = Encoding.UTF8.GetBytes(xvalue)
    ms = New IO.MemoryStream
    cs = New CryptoStream(ms, ct, _
            CryptoStreamMode.Write)
    cs.Write(byt, 0, byt.Length)
    cs.FlushFinalBlock()
    cs.Close()
    TextBox2.Text = Convert.ToBase64String _
        (ms.ToArray()) & vbCrLf & keystring & _
        vbCrLf & ivstring
    saveFile()
Else
    MessageBox.Show("No Text !", "Error", _
        MessageBoxButtons.OK)
End If
```

Keterangan Listing :

Proses yang terjadi di dalam enkripsi sebuah data sesungguhnya sama untuk berbagai jenis format file. Karena algoritma enkripsi yang digunakan di sini membutuhkan key dan IV, maka kedua unsur utama tersebut digenerate terlebih dulu dan kemudian dikonversi ke format base-64 agar dapat dimasukkan ke dalam file teks. Selanjutnya dengan berdasarkan key dan IV yang sudah ada, dilakukan proses enkripsi. Proses enkripsi secara umum akan menghasilkan sebuah rangkaian byte, tetapi agar dapat dibaca ulang di dalam textbox (dan juga terbaca sebagai file teks) maka dikonversi pula ke format base-

64. Dan langkah berikutnya adalah memanggil prosedur untuk penyimpanan file yakni *savefile* yang isinya akan dijelaskan di dalam langkah selanjutnya.

Format base-64 merupakan format string *biasa* yang dianggap sebagai format yang memenuhi kapabilitas ***human readable***. Dalam istilah yang lebih sederhana, format base-64 adalah format teks yang umumnya kita lihat di file teks. Konversi ke dalam format base-64 sering kali dilakukan agar hasil proses enkripsi ataupun proses lainnya (seperti XML Serializer) dapat terbaca ke dalam textbox atau komponen penampung teks lainnya.

6. Kini ketikkan prosedur *saveFile* yang berfungsi untuk melakukan proses penyimpanan file hasil enkripsi maupun hasil dekripsi. Prosedur ini nantinya dipanggil pada langkah terakhir proses enkripsi dan juga dekripsi.

```
Sub saveFile()
With SaveFileDialog1
    If .ShowDialog = Windows.Forms. _
        DialogResult.OK And .FileName _
        <> "" Then
        IO.File.WriteAllText _
            (.FileName, TextBox2.Text)
    End If
End With
End Sub
```

Keterangan Listing :
Dari listing tersebut, hanya dilakukan sebuah proses sederhana untuk menyimpan teks yang terdapat di dalam textbox yang kedua ke dalam sebuah file teks dengan memanfaatkan komponen savefiledialog yang tersedia.

Nantinya pada saat testing, Anda dapat membuka file teks hasil enkripsi dan melihat bahwa di dua baris terakhir hasil proses enkripsi akan terdapat key dan IV hasil generate yang disimpan dalam file yang sama. Kedua unsur utama tersebut ditandai dengan adanya karakter == di akhir barisnya. Di dalam pengembangan dari aplikasi ini, Anda dapat mengarahkan key ke sebuah hasil generate kata yang nantinya dapat diasumsikan sebagai password dari proses enkripsi.

7. Listing yang terakhir diketikkan adalah untuk button yang akan digunakan pada proses dekripsi teks.

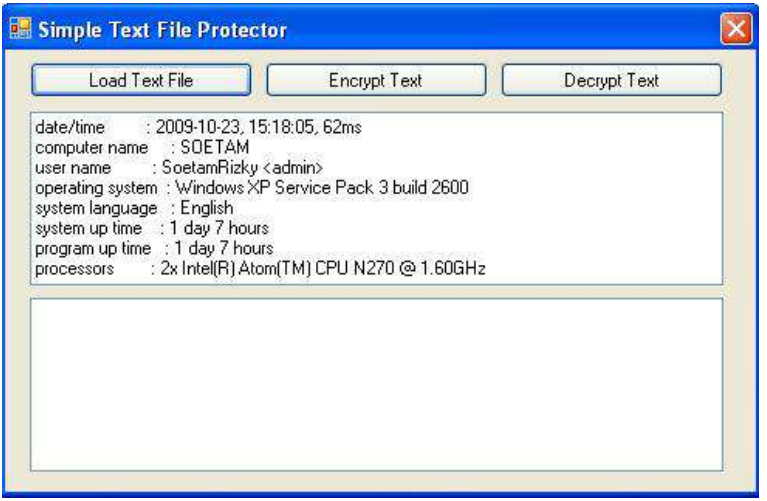
```
If TextBox1.Text <> "" Then
    Dim xtemp() As String = _
        Split(TextBox1.Text, vbCrLf)
    If UBound(xtemp) = 2 Then
        If Microsoft.VisualBasic. _
            Right(xtemp(1), 2) = "==" And _
            Microsoft.VisualBasic. _
            Right(xtemp(2), 1) = "=" Then
            Dim ct As ICryptoTransform
```

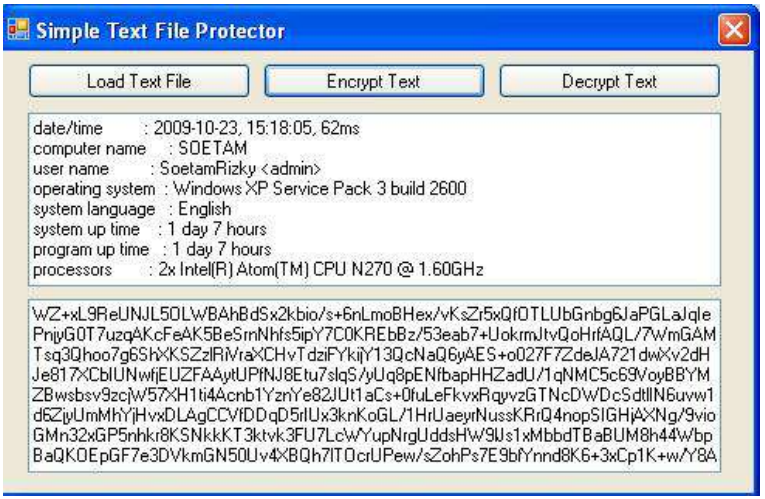
```
            Dim ms As IO.MemoryStream
            Dim cs As CryptoStream
            Dim byt() As Byte
            Dim mcsp As SymmetricAlgorithm
            Dim xvalue As String = xtemp(0)
            xkey = Convert. _
              FromBase64String(xtemp(1))
            xiv = Convert. _
              FromBase64String(xtemp(2))
            TextBox2.Clear()
            mcsp = New RC2CryptoServiceProvider
            ct = mcsp. _
               CreateDecryptor(xkey, xiv)
            byt = Convert. _
               FromBase64String(xvalue)
            ms = New IO.MemoryStream
            cs = New CryptoStream(ms, ct, _
                CryptoStreamMode.Write)
            cs.Write(byt, 0, byt.Length)
            cs.FlushFinalBlock()
            cs.Close()
            TextBox2.Text = _
                Encoding.UTF8.GetString _
                (ms.ToArray())
            saveFile()
        Else
          MessageBox.Show("No key inside !!", _
                "Error", MessageBoxButtons.OK)
        End If
    Else
        MessageBox.Show( _
            "Not a valid encrypted text !!", _
            "Error", MessageBoxButtons.OK)
    End If
Else
    MessageBox.Show( _
      "No data !!", _
         "Error", MessageBoxButtons.OK)
End If
```

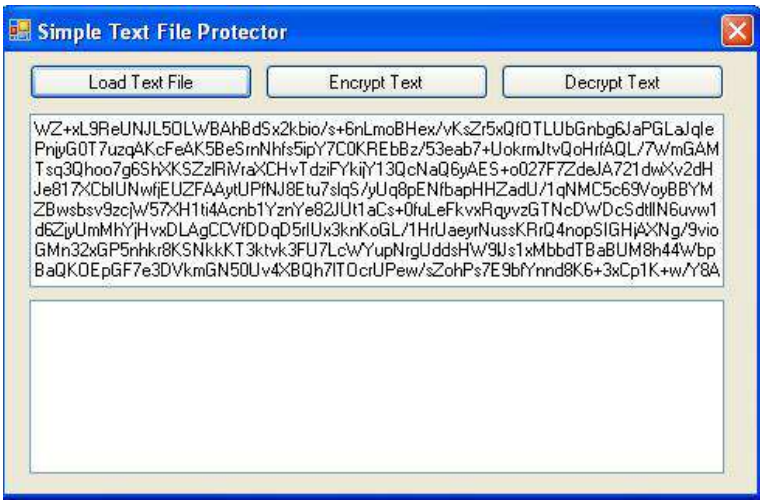
Keterangan Listing :
Di dalam proses dekripsi, sesungguhnya hanya merupakan kebalikan dari proses enkripsi. Salah satu hal terpenting di dalam proses ini adalah melakukan proses pengecekan apakah terdapat key dan IV dari proses enkripsi. Jika proses validasi tersebut berjalan dengan sukses, barulah dilakukan proses dekripsi terhadap file teks yang telah ada di dalam textbox yang pertama. Dan seperti halnya pada proses enkripsi, maka pada langkah terakhir proses, pengguna akan diarahkan untuk menyimpan hasil proses dekripsi ke dalam file dengan format teks.

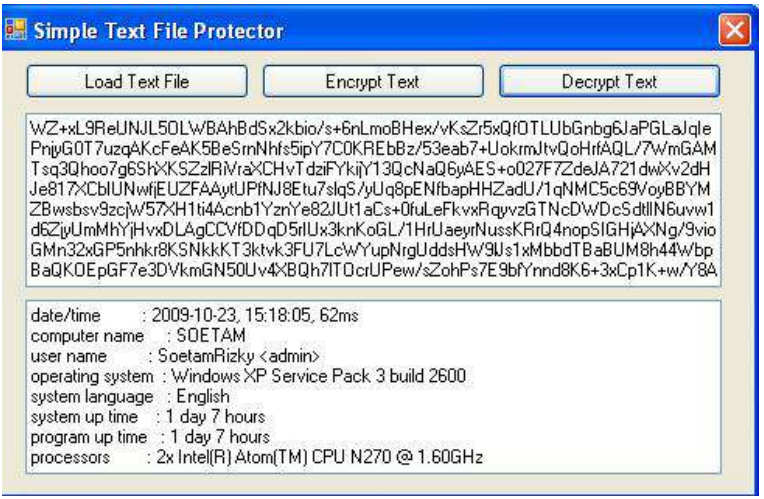
8. Kini dapat dilakukan proses running dari aplikasi tersebut. Anda dapat menggunakan berbagai jenis file teks dengan isi yang berbeda untuk melakukan proses enkripsi maupun proses dekripsi. Di dalam aplikasi sederhana ini memang terdapat sangat banyak kelemahan, karena hasil pokok dari aplikasi ini diharapkan dapat menjadi pemicu demi terciptanya aplikasi lain yang sesuai kebutuhan. Sebagai contoh dapat dikembangkan sebuah aplikasi untuk proteksi file lain yang memiliki format selain teks. Juga dapat dikembangkan agar key yang terbentuk tidak hanya dari hasil generate, misalnya key yang ada dibentuk dari *device id*

yang terdapat dalam sebuah perangkat seperti flashdisk sehingga file teks hanya akan dapat didekrip jika flashdisk yang device id-nya dijadikan sebagai kunci tertancap ke dalam perangkat komputer yang digunakan.




Simple Text File Protector
Load Text File
Encrypt Text
Decrypt Text
date/time : 2009-10-23, 15:18:05, 62ms
computer name : SOETAM
user name : SoetamRizky <admin>
operating system : Windows XP Service Pack 3 build 2600
system language : English
system up time : 1 day 7 hours
program up time : 1 day 7 hours
processors : 2x Intel(R) Atom(TM) CPU N270 @ 1.60GHz
WZ+xL9ReUNJL5OLWBAhBdSx2kbio/s+6nLmoBHex/vKsZr5xQfOTLUbGnbg6JaPGLaJqle
PnjyG0T7uzqAKcFeAK5BeSrnNhfs5ipY7C0KREbBz/53eab7+UokrmJtvQoHrfAQL/7WmGAM
Tsq3Qhoo7g6ShXKSZzlRiVraXCHvTdziFYkjY13QcNaQ6yAES+o027F7ZdeJA721dwXv2dH
Je817XCbIUNwfjEUZFAAytUPfNJ8Etu7slqS/yUq8pENfbapHHZadU/1qNMC5c69VoyBBYM
ZBwsbsv9zcjW57XH1ti4Acnb1YznYe82JUt1aCs+0fuLeFkvxRqvvzGTNcDWDcSdtlIN6uvw1
d6ZjyUmMhYjHvxDLAgCCVfDDqD5rlUx3knKoGL/1HrUaeyrNussKRrQ4nopSIGHjAXNg/9vio
GMn32xGP5nhkr8KSNkkKT3ktvk3FU7LcWYupNrgUddsHW9Us1xMbbdTBaBUM8h44Wbp
BaQKOEpGF7e3DVkmGN50Uv4XBQh7lTOcrUPew/sZohPs7E9bfYnnd8K6+3xCp1K+w/Y8A

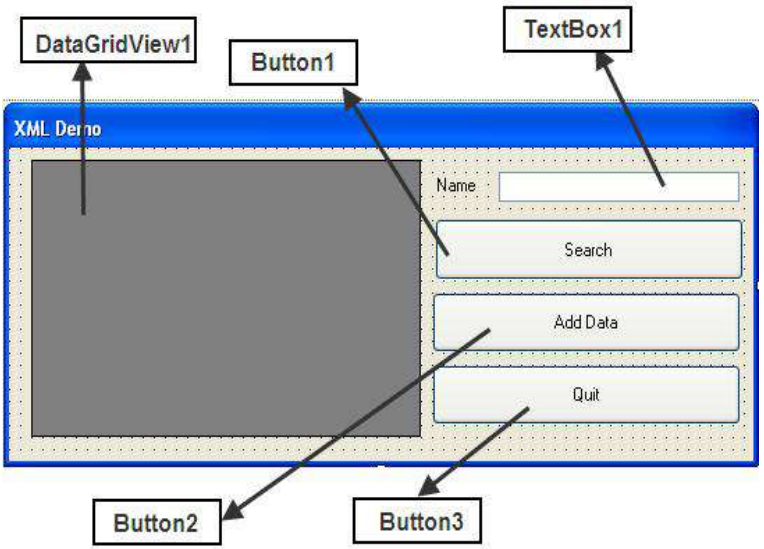

Simple Text File Protector
Load Text File
Encrypt Text
Decrypt Text
WZ+xL9ReUNJL5OLWBAhBdSx2kbio/s+6nLmoBHex/vKsZr5xQfOTLUbGnbg6JaPGLaJqle
PnjyG0T7uzqAKcFeAK5BeSrnNhfs5ipY7C0KREbBz/53eab7+UokrmJtvQoHrfAQL/7WmGAM
Tsq3Qhoo7g6ShXKSZzlRiVraXCHvTdziFYkjY13QcNaQ6yAES+o027F7ZdeJA721dwXv2dH
Je817XCbIUNwfjEUZFAAytUPfNJ8Etu7slqS/yUq8pENfbapHHZadU/1qNMC5c69VoyBBYM
ZBwsbsv9zcjW57XH1ti4Acnb1YznYe82JUt1aCs+0fuLeFkvxRqvvzGTNcDWDcSdtlIN6uvw1
d6ZjyUmMhYjHvxDLAgCCVfDDqD5rlUx3knKoGL/1HrUaeyrNussKRrQ4nopSIGHjAXNg/9vio
GMn32xGP5nhkr8KSNkkKT3ktvk3FU7LcWYupNrgUddsHW9Us1xMbbdTBaBUM8h44Wbp
BaQKOEpGF7e3DVkmGN50Uv4XBQh7lTOcrUPew/sZohPs7E9bfYnnd8K6+3xCp1K+w/Y8A

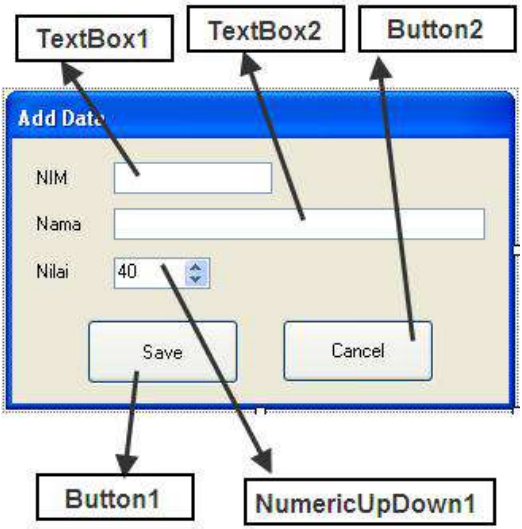

Simple Text File Protector
Load Text File
Encrypt Text
Decrypt Text
WZ+xL9ReUNJL5OLWBAhBdSx2kbio/s+6nLmoBHex/vKsZr5xQfOTLUbGnbg6JaPGLaJqle
PnjvG0T7uzqAKcFeAK5BeSmNhfs5ipY7C0KREbBz/53eab7+UokmJtvQoHrfAQL/7WmGAM
Tsq3Qhoo7g6ShXKSZzlRiVraXCHvTdziFYkiY13QcNaQ6yAES+o027F7ZdeJA721dwXv2dH
Je817XCbIUNwfjEUZFAAytUPfNJ8Etu7slqS/yUq8pENfbapHHZadU/1qNMC5c69VoyBBYM
ZBwsbsv9zcjW57XH1ti4Acnb1YznYe82JUt1aCs+0fuLeFkvxRqvvzGTNcDWDcSdtlIN6uvw1
d6ZjvUmMhYjHvxDLAgCCVfDDqD5rlUx3knKoGL/1HrUaeyrNussKRrQ4nopSIGHjAXNg/9vio
GMn32xGP5nhkr8KSNkkKT3ktvk3FU7LcWYupNrgUddsHW9Us1xMbbdTBaBUM8h44Wbp
BaQKOEpGF7e3DVkmGN50Uv4XBQh7ITOcrUPew/sZohPs7E9bfYnnd8K6+3xCp1K+w/Y8A
date/time : 2009-10-23, 15:18:05, 62ms
computer name : SOETAM
user name : SoetamRizky <admin>
operating system : Windows XP Service Pack 3 build 2600
system language : English
system up time : 1 day 7 hours
program up time : 1 day 7 hours
processors : 2x Intel(R) Atom(TM) CPU N270 @ 1.60GHz

Contoh 4 : XML Editor Sample

Level : Mahir
Tujuan :
1. Mengenal manipulasi file XML sebagai dataset
2. Mengenal struktur file XML dan penggunaannya dalam sebuah aplikasi
3. Mengenal akses variabel dan prosedur lintas form dengan menggunakan *scope* public.
4. Mengenal penggunaan komponen DataGridView

Desain awal :


DataGridView1
Button1
TextBox1
XML Demo
Name
Search
Add Data
Quit
Button2
Button3


TextBox1
TextBox2
Button2
NIM
Nama
Nilai
40
Save
Cancel
Button1
NumericUpDown1

Langkah Penyelesaian :

1. Buat sebuah solution baru yang didalamnya terdiri dari dua buah form. Form yang pertama ganti property name menjadi *frmMain*, sedangkan nama dari form yang kedua adalah *AddData*. Form pertama berfungsi sebagai tampilan utama dari data yang ditampilkan dari file XML, sedangkan form yang kedua nantinya akan digunakan untuk menambah data di dalam file XML yang terbentuk pada saat form dijalankan.
2. Di dalam form yang pertama, tempatkan komponen-komponen berikut ini dengan susunan tata letak seperti pada gambar :
    a. Sebuah datagridview dengan property *ReadOnly* diset nilainya menjadi *true*.
    b. Sebuah label dengan property *Text* yang berisikan nilai *Name*
    c. Sebuah textbox di sebelah label.
    d. Tiga buah button yang masing-masing property *Text*-nya diisi dengan teks *Search, Add Data* dan *Quit*.

Komponen datagridview utamanya berfungsi untuk menampilkan data yang berasal dari sebuah tabel di dalam database. Tetapi di dalam contoh ini, demi memperkenalkan kegunaan datagridview, akan diisikan data yang berasal dari sebuah file XML. Dan untuk mempermudah pengenalan terhadap file XML, maka file XML yang akan diakses sebelumnya diarahkan ke sebuah variabel obyek dataset. Sehingga pengelolaan file XML nantinya akan menyerupai pengelolaan sebuah tabel di dalam database. Tapi tetap perlu diperhatikan bahwa file XML sendiri bukanlah database dan memang tidak ditujukan untuk menggantikan peran dari database.

3. Sedangkan untuk form yang kedua tempatkan komponen-komponen berikut ini :
    a. Tiga buah label dengan isi property *Text* masing-masing adalah *NIM, Name* dan *Score*.
    b. Dua buah textbox
    c. Sebuah komponen numericupdown dengan setting property *Minimum* yang diganti menjadi angka 40, begitu pula dengan isi dari property *Value*.

d. Dua buah button dengan isi property *Text* masing-masing adalah *Save* dan *Cancel*.
e. Sedangkan untuk property *TopMost* dari form diset menjadi *true*, dan property *ControlBox* diset menjadi *false*.

Karena form yang kedua lebih berfungsi untuk form sekunder, maka diharapkan agar pada saat form kedua tampil, maka form pertama tidak dapat diakses oleh pengguna. Hal ini lazim dilakukan di sebuah aplikasi database yang kompleks (dapat Anda temui di buku jilid kedua) agar proses yang dilakukan benar-benar terpisah.

4. Selanjutnya, beralihlah ke code view di form yang pertama dan tepat di bagian yang paling atas, ketikkan listing berikut :

```
Imports System.Xml
```

Keterangan Listing :
Ini berarti bahwa di dalam listing form pertama nantinya akan menggunakan sintaks-sintaks yang berada di dalam library XML.

5. Berikutnya di bawah deklarasi *Public Class…* (masih di code view form yang pertama), ketikkan deklarasi variabel-variabel berikut ini.

```
Public Shared xset As _
      New Data.DataSet
Public Shared fileXML As _
      String = "Mahasiswa.xml"
Public Shared xtabel As _
      New DataTable("DataMahasiswa")
```

Keterangan Listing :
Seluruh variabel yang dideklarasikan tersebut nantinya juga akan diakses oleh form kedua atau form *AddData*. Itulah sebabnya ketiganya diset dengan *scope* public. Variabel yang pertama berfungsi untuk membentuk obyek dataset yang akan digunakan sebagai penampung dari pembacaan file XML. Sedangkan variabel yang kedua merupakan nama file XML yang akan dijadikan ajang percobaan dalam aplikasi ini. Dan variabel yang terakhir adalah variabel untuk menampung obyek datatable hasil dari generate dataset sebelumnya.

6. Kini dobel klik di bagian kosong pada form pertama sehingga menuju ke prosedur *Form_Load* dan ketikkan listing didalamnya.

```
Dim tabulasi As New XmlWriterSettings()
With tabulasi
    .Indent = True
    .IndentChars = Space(5)
End With
Using xWriter As XmlWriter = _
    XmlWriter.Create(fileXML, tabulasi)
    With xWriter
        Dim xNama() As String = _
            Split("Ali,Baba,Caca,Didi,Esa," & _
                "Fafa,Gago", ",")
        .WriteStartElement("DataMahasiswa")
       For i As Integer = 0 To xNama.Length - 1
```

```
            .WriteStartElement("Mahasiswa")
     .WriteAttributeString("NIM", "00" & i + 1)
        .WriteAttributeString("Nama", xNama(i))
            .WriteAttributeString("Nilai", _
                  CInt(Rnd() * 100) + 20)
            .WriteEndElement()
        Next
        .WriteEndElement()
        .Flush()
    End With
End Using
readXML()
```

Keterangan Listing :

Pada saat form pertama dijalankan pertama kali, maka akan dibuat sebuah file XML dengan nama *Mahasiswa.Xml*. Isi dari file XML ini terdiri dari susunan satu buah *root element* bernama *DataMahasiswa*. Kemudian terdiri dari satu susunan kerangka *child element* dengan nama *Mahasiswa*. Sedangkan untuk melakukan simulasi field dari sebuah tabel, maka dibentuk *attribute* dalam file XML tersebut sebanyak tiga yakni *NIM, Nama* dan *Nilai*. Untuk isian data awal terdapat tujuh data yang digenerate, *NIM* diisi dengan urutan dari 001 hingga 007. Sedangkan atribut *Nama* diisi melalui array *xNama* secara berurutan. Dan untuk atribut yang terakhir yaitu *Nilai* diisi dengan angka acak yang memiliki range dari 40 hingga angka 100 sebagai batas tertinggi. Di akhir pengisian data pada file XML, terdapat perintah *Flush* yang berarti bahwa pengisian tersebut diperintahkan untuk segera dituliskan ke dalam file dari *buffer* yang digunakan sebagai penampung sementara. Setelah pengisian data berakhir, maka selanjutnya dipanggil prosedur *readXML* yang merupakan prosedur untuk menampilkan file XML ke dalam datagridview. Isi dari prosedur tersebut dapat dilihat di langkah selanjutnya.

7. Langkah berikutnya adalah mengetikkan listing untuk prosedur *readXML*.

```
Public Shared Sub readXML()
xset.Clear()
xset.ReadXml(fileXML, _
    XmlReadMode.Auto)
xtabel = xset.Tables(0)
frmMain.DataGridView1.DataSource = _
    xset.Tables(0)
frmMain.DataGridView1.Refresh()
End Sub
```

Keterangan Listing :
Secara umum, cara pengisian data ke dalam datagridview sangatlah sederhana. Dengan memasukkan data yang dimaksud ke dalam sebuah obyek dataset, maka datagridview akan langsung mampu membaca data di dalam obyek dataset tersebut. Konsep sederhana dari obyek dataset adalah diibaratkan sebagai sebuah penampung database sementara secara virtual yang didalamnya juga dapat berisi tabel secara majemuk. Dalam contoh ini, dataset diisi sebuah tabel yang sebenarnya adalah pengejawantahan dari struktur file XML yang sebelumnya dibuat pada saat form pertama kali dijalankan.

Untuk mempelajari tentang XML lebih lanjut Anda dapat membaca buku *Sistem Informasi Terintegrasi dengan menggunakan XML Web Service* dari pengarang yang sama.

8. Kemudian dobel klik button *Search* untuk mengetikkan listing berikut ini :

```
Dim xkey(0) As DataColumn
xkey(0) = xset.Tables(0).Columns(1)
xset.Tables(0).PrimaryKey = xkey
Dim xrow As DataRow = _
    xset.Tables(0).Rows. _
    Find(TextBox1.Text)
If IsNothing(xrow) Then
    MsgBox("Tidak ditemukan !")
Else
    MsgBox("Nilai : " & _
           xrow.Item("Nilai"))
End If
```

Keterangan Listing :

Pada saat pencarian data, dengan menggunakan konsep dataset, maka dapat didefinisikan sebuah atau lebih field imitasi sebagai *primary key*. Dalam contoh ini, field imitasi yang berasal dari atribut file XML dan dijadikan sebagai primary key adalah field *Nama*. Setelah primary key temporer tersebut dibuat, maka untuk melakukan pencarian dilakukan dengan menggunakan perintah *Find*. Dan untuk mendeteksi apakah field yang dicari ada dalam datatable, dapat dicek dengan melihat apakah ada *row* atau record yang muncul dari hasil pencarian tersebut. Bila ternyata ditemukan, maka ditampilkan message box yang membuktikan bahwa field tersebut memang berhasil dicari. Pencarian data di dalam sebuah file XML sendiri memiliki banyak varian teknik, sebagai contoh adalah pencarian dengan memanfaatkan XPATH ataupun Xquery. Bahkan di .NET Framework 3.5 juga terdapat fasilitas LINQ (Language Integrated Query) yang dapat mensimulasikan penggunaan perintah SQL dalam pencarian data di sebuah file XML. Pemilihan metode pencarian sangat bergantung pada proses pembentukan sekaligus manipulasi yang dilakukan pada file XML itu sendiri.

Konsep primary key dan manipulasi dataset secara utuh dapat Anda temui di buku yang memiliki judul sama di jilid yang kedua dari pengarang yang sama pula.

9. Untuk button *Add Data* ketikkan listing berikut ini yang berfungsi untuk memanggil form kedua agar muncul dengan mode *Modal*.

```
AddData.ShowDialog()
```

10. Listing terakhir di form yang pertama adalah untuk button *Quit* yang berguna untuk keluar dari aplikasi secara utuh.

```
Application.Exit()
```

Di dalam sebuah aplikasi form majemuk, akan terlihat sangat jelas perbedaan antara *close* dan *exit*. Close digunakan untuk menutup sebuah form, sedangkan exit digunakan untuk menutup aplikasi seccara keseluruhan.

11. Kini saatnya beralih ke form yang kedua atau form *AddData*. Dobel klik di button untuk menyimpan data atau button *Save* lalu ketikkan listing berikut ini :

```
With frmMain
    Dim xrow As DataRow = .xtabel.NewRow
```

```
    xrow("NIM") = TextBox1.Text
    xrow("Nama") = TextBox2.Text
    xrow("Nilai") = NumericUpDown1.Value
    .xtabel.Rows.Add(xrow)
    .xset.Tables(0).WriteXml(.fileXML)
    .readXML()
End With
Close()
```

Keterangan Listing :

Di form yang kedua, tepatnya di listing untuk penyimpanan data ke dalam file XML, ketiga variabel yang ada di dalam form pertama dipanggil di sini. Begitu pula dengan prosedur *readXML* yang ada di dalam form pertama. Jika diperhatikan pada saat pembuatan prosedur *readXML*, pada saat perintah untuk mengakses datagridview terdapat pengenal *frmMain* yang menunjukkan bahwa prosedur tersebut merupakan prosedur yang siap diakses dari form lain selama berada dalam solution yang sama.

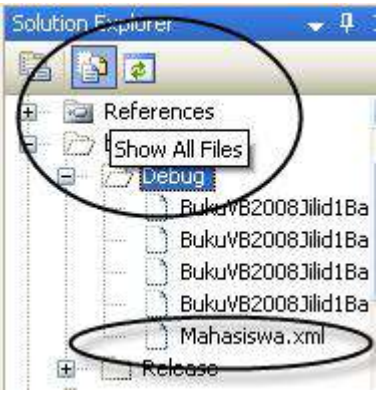
Sedangkan pada saat proses penyimpanan data ke dalam file XML, manipulasi dengan menggunakan media dataset akan sangat memudahkan proses. Dapat dilihat bahwa ketiga atribut yang akan diisikan dianggap sebagai field-field dalam sebuah tabel sehingga cara penambahannya menafikan struktur tree dari sebuah file XML yang seringkali menjadi penghambat dalam proses pembelajaran teori dan konsep file XML.

Setelah proses penyimpanan selesai, prosedur *readXML* dipanggil ulang agar datagridview yang terletak di form pertama melakukan refresh dengan isian data yang baru. Dalam proses penyimpanan ini memang belum terdapat validasi data (isian kosong, data yang valid) sehingga dalam proses pengembangannya dapat ditambahkan beberapa tambahan seperti validasi data isian dari textbox.

12. Listing terakhir adalah di button *Cancel* yang akan digunakan untuk menutup form kedua dan kembali ke form yang pertama tanpa melakukan penyimpanan data.

```
Close()
```

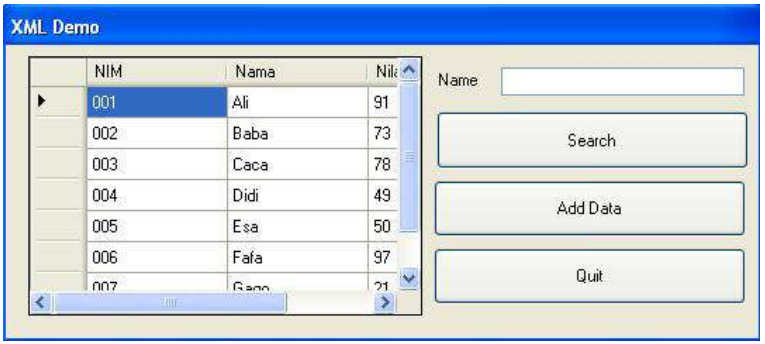
13. Pada saat aplikasi ini dijalankan, dapat dilihat di dalam Solution Explorer bahwa sebuah file XML akan terbentuk di dalam folder tempat aplikasi dikompilasi. Setelah aplikasi selesai dirunning, maka di dalam Solution Explorer dapat diklik icon *Show All Files*, lalu arahkan ke dalam folder *bin* **à** *Debug*. Jika aplikasi sudah benar dijalankan, maka akan terlihat sebuah file baru bernama *Mahasiswa.XML* yang merupakan hasil dari proses generate file XML di form pertama.



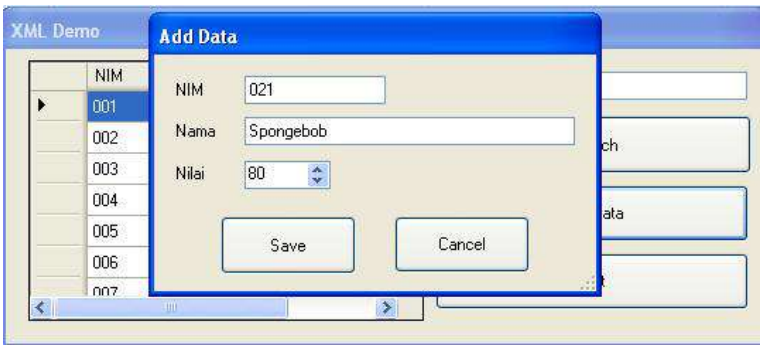
14. Jika Anda melakukan dobel klik pada file tersebut, maka di dalam lingkup IDE Visual Studio akan tampak isi dari file XML yang digenerate tersebut.

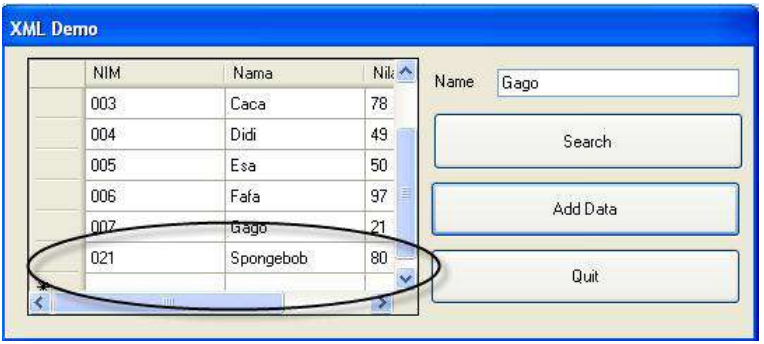
```
<?xml version="1.0" standalone="yes"?>
<DataMahasiswa>
  <Mahasiswa NIM="001" Nama="Ali" Nilai="91" />
 <Mahasiswa NIM="002" Nama="Baba" Nilai="73" />
 <Mahasiswa NIM="003" Nama="Caca" Nilai="78" />
 <Mahasiswa NIM="004" Nama="Didi" Nilai="49" />
  <Mahasiswa NIM="005" Nama="Esa" Nilai="50" />
 <Mahasiswa NIM="006" Nama="Fafa" Nilai="97" />
 <Mahasiswa NIM="007" Nama="Gago" Nilai="21" />
</DataMahasiswa>
```

15. Berikut ini adalah hasil contoh proses editor XML sederhana dengan menggunakan teknik dataset.


XML Demo
NIM
Nama
001
Ali
91
002
Baba
73
003
Caca
78
004
Didi
49
005
Esa
50
006
Fafa
97
Name
Search
Add Data
Quit


XML Demo
NIM
Nama
001
Ali
91
Name
Gago
BukuVB2008Jilid1BabLanjutan
Nilai : 21
OK


XML Demo
Add Data
NIM
021
Nama
Spongebob
Nilai
80
Save
Cancel


XML Demo
NIM
Nama
Nil
003
Caca
78
004
Didi
49
005
Esa
50
006
Fafa
97
007
Gago
021
Spongebob
80
Name
Gago
Search
Add Data
Quit

Appendix – Pengantar .NET Framework

Sekilas Sejarah Visual Basic

Jika dirunut ke belakang, Visual Basic berasal dari bahasa BASIC (Beginner's All-purpose Symbolic Instruction Code) yang dianggap sebagai awal mula keberhasilan pembelajaran bahasa pemrograman bagi para pemula secara mudah dan cepat. Selanjutnya dengan berbagai macam variannya seperti GWBASIC, Turbo Basic dan BASICA, pada tahun 1991, Microsoft mengeluarkan Visual Basic 1.0 yang kemudian dianggap sebagai bahasa pemrograman berbasis RAD (Rapid Application Development) yang paling mudah dipelajari saat itu.

Berikutnya, Visual Basic semakin berkembang, sehingga pada tahun berikutnya keluar versi 2.0, dan versi 3.0 juga keluar pada tahun berikutnya. Lompatan terbesar dan dianggap sebagai versi paling stabil (sekaligus versi terakhir dari Visual Basic) adalah versi 6.0 yang secara resmi diluncurkan pada tahun 1998. Visual Basic .NET sendiri secara resmi dikeluarkan

pada tahun 2002, dan merupakan bagian dari .NET Framework.

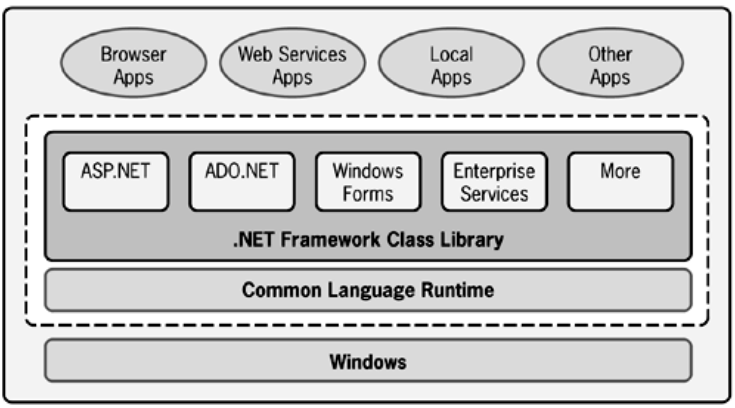
Sekilas .NET Framework

Satu hal yang sangat penting diingat adalah bahwa Visual Basic .NET merupakan bagian dari .NET Framework. .NET Framework sendiri dapat diartikan (dalam istilah yang lebih sederhana) sebagai sebuah kerangka yang mendukung pengembangan perangkat lunak serta menjadi bagian yang terintegrasi dalam sistem operasi Windows. .NET Framework sendiri tidak hanya menaungi Visual Basic .NET, tetapi juga bahasa pemrograman lain yang didukung oleh .NET Framework seperti Visual C#, Visual J#, Delphi .NET, Iron Phyton dan masih banyak lagi.

.NET Framework sendiri terintegrasi ke dalam sistem operasi Windows (sesuai dengan versi tiap Windows). Dalam Windows XP SP2 telah terintegrasi .NET Framework versi 1.0, begitu pula dengan Windows 2000 dengan SP4 serta Windows Server 2003.

Bagian dasar dari .NET Framework yang penting adalah CLR (Common Language Runtime). Saat sebuah aplikasi yang dibuat dengan

menggunakan Visual Basic .NET dijalankan, maka bagian MSIL (Microsoft Intermediate Language) dari CLR yang melakukan kompilasi terhadap aplikasi. Selain CLR, bagian dasar dari .NET Framework adalah .NET Framework Class Library yang menyediakan banyak hal dalam pengembangan aplikasi seperti ADO .NET serta ASP .NET.



Skema .NET Framework

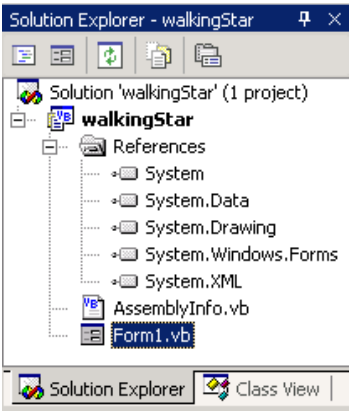
Dengan kata lain, saat sebuah aplikasi menggunakan Visual Basic .NET sebagai bahasa pemrogramannya, maka aplikasi tersebut tidak hanya terbatas pada aplikasi berbasis desktop,

tetapi juga dapat berupa aplikasi berbasis web (menggunakan ASP .NET yang berbahasa Visual Basic .NET).

Appendix – Pengantar IDE Visual Basic 2008 Express Edition

Lingkungan Visual Basic .NET merupakan sebuah IDE (Integrated Development Environment) yang terpadu dengan Visual Studio .NET. Bagian – bagian dari sebuah lingkungan Visual Basic .NET adalah sebagai berikut :

Solution Explorer

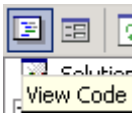


Solution Explorer

Solution explorer merupakan window Visual Basic .NET yang memperlihatkan solution serta bagian – bagian yang ada didalamnya. Solution explorer mempunyai sebuah *treeview* yang menampilkan beberapa item standar seperti : nama project, nama form, nama modul ataupun nama class serta referensi dari sebuah project itu sendiri.

Anda bisa menggunakan icon Auto Hide untuk memperluas area kerja yang dimiliki. Sehingga tiap window dapat menyembunyikan dirinya sendiri secara otomatis.

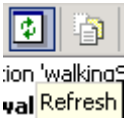
Solution explorer juga berguna untuk mengakses bagian – bagian dari sebuah project secara cepat dengan mengakses icon yang terdapat di bagian atas. Icon tersebut adalah :

1. 

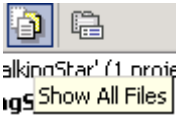
Berfungsi untuk menuju ke code editor sebuah form atau module

2. 

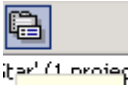
Berfungsi untuk menuju ke windows form designer

3. 

Berfungsi untuk melakukan refresh solution explorer setelah terjadi modifikasi di dalam sebuah solution

4. 

Berfungsi untuk menampilkan semua files yang terdapat dalam solution

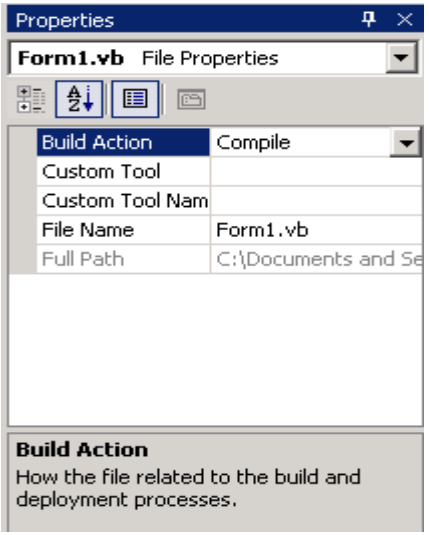
5. 

Berfungsi untuk menuju ke properties window

Properties Window

Properties window adalah window yang berisi properti dari sebuah control atau sebuah item di dalam project (module, form atau class) secara *design time*. Di dalam sebuah properties window terdapat icon – icon sebagai berikut :

1.  Categorized Merupakan icon yang berfungsi untuk mengelompokkan properties berdasarkan kategori di kontrol tersebut.
2.  Alphabetic Merupakan icon yang berfungsi untuk mengelompokkan properties berdasarkan alfabet. Cara pengurutan ini cenderung lebih memudahkan bagi para programmer, karena programmer tidak perlu mengetahui kategori dari sebuah property, hanya cukup mencari dengan pedoman huruf awal dari sebuah property.



Properties Window

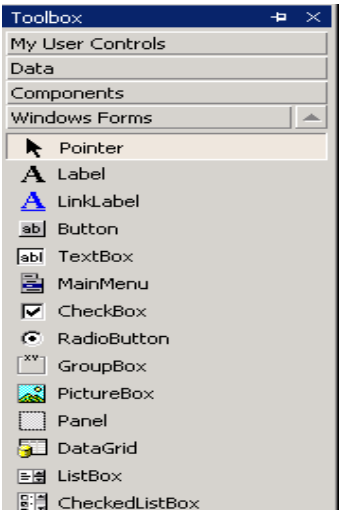
Properties window akan hilang jika Anda sedang berada di dalam mode pengeditan listing program atau sedang dalam code editor

Windows Form Designer dan Code Editor

Windows form designer adalah lingkungan tempat mendesain sebuah form atau sebuah user control. Di dalam windows form designer umumnya terdapat sebuah form yang dapat ditempati oleh banyak komponen didalamnya.

Code editor adalah tempat untuk mengetikkan kode program di dalam Visual Basic .NET. Code editor dapat diakses melalui solution explorer ataupun dengan melakukan klik kanan dan klik ganda pada windows form designer.

Toolbox

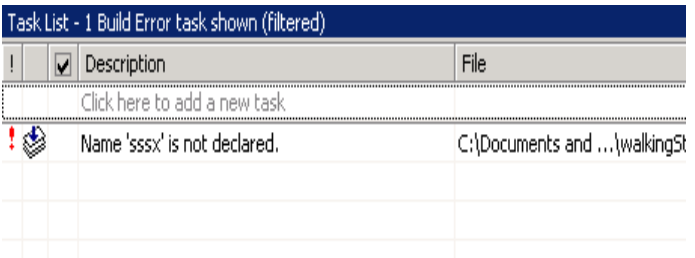


Toolbox

Toolbox adalah tempat icon – icon dari komponen yang ada dalam Visual Basic .NET ditempatkan. Jika posisi berada pada code editor, maka secara otomatis, toolbox akan menjadi *disable*, tetapi pada saat berada di windows form designer, toolbox baru akan aktif. Tab dari toolbox juga bervariasi berdasarkan aplikasi yang sedang dirancang. Sebagai contoh : pada saat perancangan aplikasi dengan menggunakan ASP

.NET, toolbox akan memunculkan tab baru untuk web form

Task List



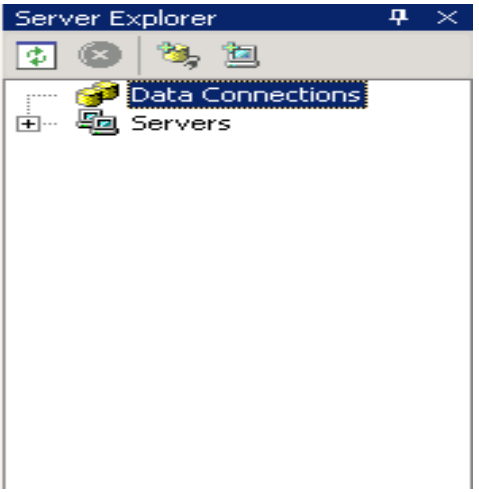
Tasklist

Tasklist merupakan salah satu window terpenting yang perlu diketahui oleh seorang programmer di lingkungan Visual Basic .NET. Secara detail, tasklist memiliki banyak fungsi yang sangat membantu dalam proses pemrograman. Tetapi fungsi terpenting sekaligus yang paling sering ditemui dari tasklist adalah sebagai penampung pesan kesalahan jika terjadi *error* dari aplikasi yang sedang dieksekusi.

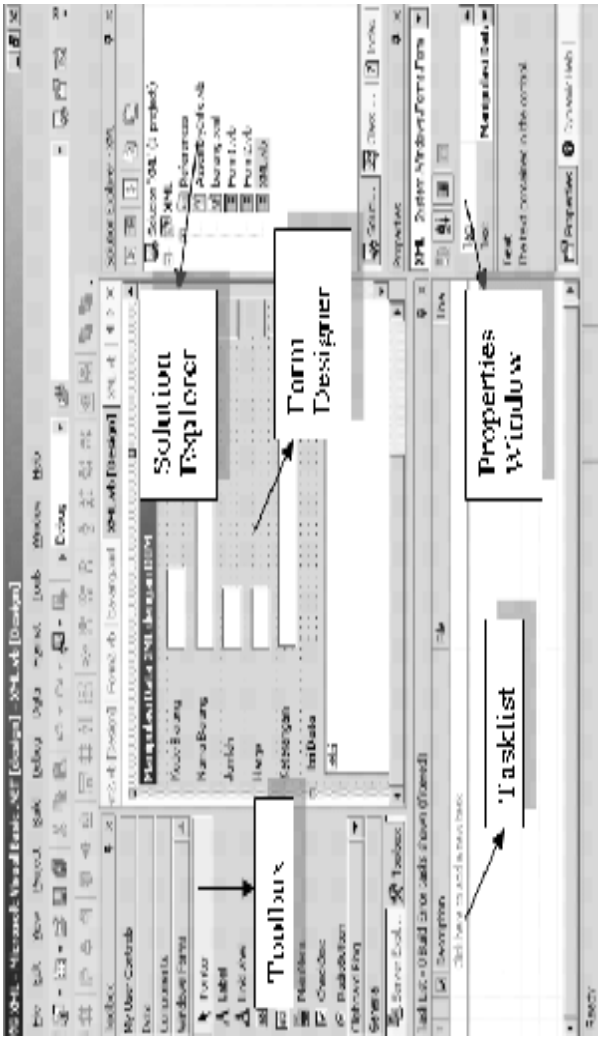
Server Explorer

Server explorer adalah window khusus yang mampu melakukan koneksi dengan server database seperti Oracle, SQL Server atau database desktop seperti Access. Server explorer merupakan alat bantu baru di lingkungan Visual Basic .NET yang sangat membantu dalam pembuatan aplikasi database. Sebab, kemampuan dari server explorer telah hampir menggantikan kemampuan dari database manager itu sendiri, seperti menampilkan database, struktur tabel, isi data hingga ke eksekusi stored procedures.

Dengan bantuan server explorer, saat terjadi pemrograman untuk aplikasi database, seorang programmer akan menghemat waktunya dengan hanya membuka server explorer dibanding dengan membuka interface database manager yang digunakan misal : SQL Server dengan Enterprise Managernya.



Server Explorer



Lingkungan Visual Basic .NET

Appendix - Struktur Aplikasi Visual Basic

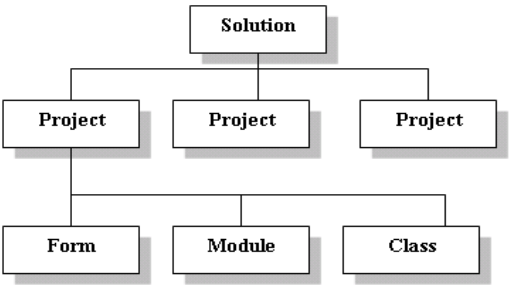
Struktur sebuah aplikasi di Visual Basic .NET dimulai dari sebuah *solution* yang bisa terdiri dari berbagai jenis project dari berbagai jenis bahasa pemrograman.

Umumnya para programmer sering rancu saat membuat sebuah project baru, padahal umumnya yang dilakukan adalah membuat sebuah solution baru.

Sebuah aplikasi di Visual Basic .NET adalah *folder based application* yang secara otomatis akan menempatkan semua solution, project serta semua item yang ada didalamnya dalam sebuah folder tertentu yang namanya sama dengan nama solution yang dibuat. Hal itu juga terjadi saat proses *build* dan *run* terjadi, akan langsung terbentuk file hasil kompilasi dalam sebuah sub folder baru di folder yang sama. Sehingga, programmer akan lebih mudah mengidentifikasi serta melacak keberadaan semua item yang ada dalam sebuah project.

Secara umum pula, semua ekstensi file yang menjadi item di dalam sebuah project di Visual Basic .NET akan sama yaitu *.vb*. Hal ini akan

memudahkan pengenalan file, tetapi di sisi lain, tetapi bagi Anda, para veteran Visual Basic 6.0 akan cukup menjadi sebuah hal baru yang sedikit membingungkan.



Struktur aplikasi

Solution dan Project

Dalam implementasinya, seringkali programmer Visual Basic .NET rancu akan istilah solution dan project. Hal ini dikarenakan setiap kali terjadi pembuatan sebuah project yang benar – benar baru akan secara otomatis membentuk pula sebuah solution baru.

Sebuah solution adalah hirarki tertinggi dalam pembuatan aplikasi di Visual Basic .NET dan akan membentuk dua buah file yaitu :

1. File dengan ekstensi *.sln* yang berisikan informasi tentang :
    a. Project – project yang ada dalam solution tersebut.
    b. Item lain yang tidak ada hubungannya dengan project tetapi termasuk dalam solution yang bersangkutan
    c. Konfigurasi *build* dalam setiap project yang ada didalamnya.
2. File yang berekstensi *.suo* yang berisikan konfigurasi IDE saat sebuah solution dikerjakan. Hal ini akan membantu

programmer, terutama jika mengerjakan sebuah aplikasi di sebuah komputer yang mempunyai banyak pemakai. Sehingga tiap programmer dapat memiliki susunan window sendiri sesuai dengan kebutuhan masing - masing

Project

Sebuah project merupakan bagian utama dari sebuah aplikasi dalam Visual Basic .NET. Sebuah project dapat berisikan :

1. Physical

   Berupa file – file yang termasuk di dalam project seperti form, class library ataupun module. Juga didalamnya adalah file project itu sendiri serta folder yang secara otomatis terbentuk saat kita membangun sebuah project baru.

2. Virtual

   Yang dimaksud dengan item virtual adalah representasi dari pointer yang menunjuk ke sebuah item fisik, misal : koneksi database, virtual direktori (untuk aplikasi web) dan lainnya.

Di dalam Visual Basic .NET terdapat berbagai macam *templates* project diantaranya :

1. Windows application template

   Ini merupakan template default dari project di Visual Basic .NET. Template ini

setara dengan *Standard project EXE* di Visual Basic 6.0. Secara keseluruhan, buku ini lebih banyak menggunakan template pertama ini dalam contoh soal latihannya.

Hampir semua contoh project yang terdapat dalam buku ini merupakan project yang menggunakan Windows application template.

2. Class library template

   Template ini digunakan untuk pembuatan class ataupun komponen yang dapat dipakai di project yang berbeda.

3. Windows control library template

   Template ini setara dengan project *user control* di Visual Basic 6.0. Umumnya digunakan untuk membuat sebuah komponen turunan dari komponen yang sudah ada sebelumnya. Komponen turunan tersebut bisa mempunyai sifat – sifat yang baru atau juga bisa merupakan gabungan dari beberapa komponen yang dijadikan satu.

4. ASP.NET web application template

Template ini akan membangun sebuah aplikasi web dengan menggunakan ASP.NET. Karena pada Visual Basic .NET, ASP .NET telah menjadi sebuah bagian yang terintegrasi di dalam Visual Basic .NET itu sendiri. Pembuatan aplikasi dengan menggunakan template ini membutuhkan instalasi IIS (Internet Information Services). Sebab, saat pertama kali dibuat, Visual Basic .NET akan langsung mengakses IIS serta membuat sebuah direktori virtual di dalam web server yang kita miliki. Akibatnya, jika kita membuat aplikasi dengan template ini di sebuah komputer dengan sistem operasi Windows versi 2000 ke atas, kita harus memiliki hak akses yang cukup agar dapat memasuki IIS serta membuat direktori virtual tersebut.

5. ASP.NET mobile web application template
   Sama halnya dengan template sebelumnya, tetapi lebih dikhususkan untuk aplikasi yang dijalankan pada mobile devices seperti PDA.
6. ASP.NET web service template

Web service merupakan kekuatan baru di dalam Visual Basic .NET. Pembuatan web service sebenarnya akan membangun sebuah XML web service yang dapat digunakan dalam aplikasi lain yang berada dalam satu jaringan.

7. Web control library template

   Sama seperti windows control library template, tetapi lebih dikhususkan untuk aplikasi web dengan menggunakan ASP.NET.

8. Console application template

   Console application merupakan aplikasi yang tidak memiliki tampilan secara grafis. Aplikasi ini dijalankan dalam sebuah command line dengan parameter –parameter tertentu.

9. Windows service template

   Digunakan untuk membangun aplikasi yang dijalankan sebagai sebuah *services* di sistem operasi Windows. Dalam aplikasi gaya lama seringkali dianalogikan dengan aplikasi TSR (terminate and stay resident) artinya aplikasi yang biasanya dijalankan selama Windows

masih melakukan proses dan tidak terlihat oleh pengguna secara eksplisit.

10. New project in existing folder template

    Tipe ini biasanya digunakan untuk menambahkan sebuah project baru ke dalam sebuah folder yang telah ada sebelumnya.

11. Empty project template
12. Empty web project template

Selain itu, juga terdapat beberapa jenis template yang ditujukan untuk semua bahasa pemrograman yang ada dalam Visual Studio .NET, tetapi juga sering digunakan di dalam Visual Basic .NET, yaitu :

1. Deployment project

   Digunakan untuk membuat aplikasi instalasi yang digunakan untuk mendistribusikan aplikasi Visual Basic .NET ke komputer lain yang tidak terinstalasi Visual Basic .NET .

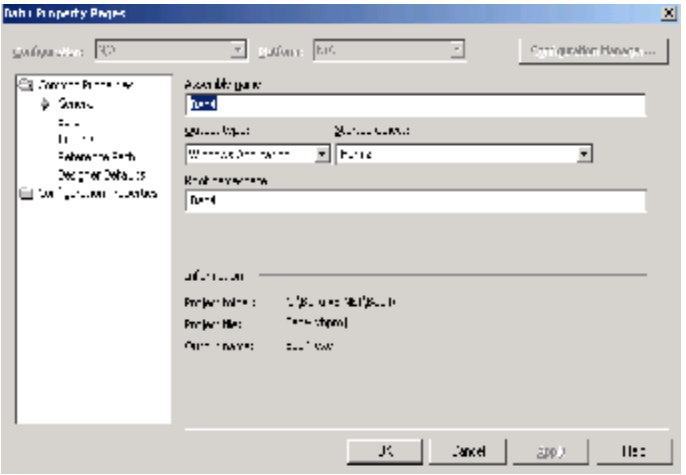
2. Database project

   Digunakan untuk membuat script – script baru di sebuah database server seperti SQL Server

ataupun Oracle yang nantinya akan diikutsertakan dalam sebuah aplikasi lain.

Property Project

Sebuah project standar yang dimaksud di dalam buku ini adalah project yang menggunakan template windows application template. Property sebuah project standar dapat dilihat dengan menggunakan menu Project dan di sub menu properties, atau juga dengan melakukan klik kanan pada nama project di solution explorer dan memilih properties.



Kotak dialog project property

Property project terdiri dari dua bagian yaitu :

1. Common

2. Configuration

Property

Property seringkali dianalogikan dengan sifat sebuah benda. Di dalam sebuah pemrograman visual khususnya di Visual Basic .NET, property merupakan sifat – sifat yang dimiliki oleh sebuah obyek, baik itu berupa obyek yang berasal dari class yang dibuat oleh programmer ataupun dari obyek yang berbentuk komponen seperti textbox, button dan lainnya.

Property mampu mengatur cara penempatan dan metode tampilan sebuah komponen untuk pengguna, selain itu property juga mampu melakukan pengaturan untuk cara interaksi antara pengguna dengan aplikasi.

Pengaturan property dapat dilakukan dengan dua macam cara :

1. Design time

   Pengaturan property dilakukan saat aplikasi belum dieksekusi. Umumnya pengaturan saat design time dilakukan dengan cara mengakses property melalui property window. Saat design time, hampir dipastikan pengaturan property

tiak akan salah, karena efek yang ditimbulkan dapat langsung dilihat oleh programmer, mis : pengaturan property untuk *backcolor* untuk sebuah akan langsung terlihat oleh programmer di windows form designer.

3. Run time

   Pengaturan saat run time berarti melakukan pengaturan melalui listing program. Resiko yang ditimbulkan adalah kesalahan yang mungkin terjadi saat aplikasi dijalankan, karena programmer belum melihat efek dari perubahan property hingga aplikasi tersebut dieksekusi. Tetapi, terdapat beberapa property yang hanya bisa diakses melalui run time, misalnya : property *left* yang terdapat di komponen textbox.

Methods

Methods adalah prosedur yang diasosiasikan ke sebuah object atau seringkali ke sebuah komponen. Sebuah methods seringkali dianalogikan sebagai sesuatu yang bisa dilakukan oleh sebuah object. Misalnya sebuah form di Visual Basic .NET mampu menempatkan dirinya ke tengah – tengah layar secara tepat, jika diberikan method *CenterToScreen* untuk form tersebut.

Sebuah komponen dapat memiliki lebih dari satu method, dan beberapa method dengan nama serta fungsi yang sama dapat dijumpai dalam berbagai komponen. Dengan adanya kesamaan tersebut akan memudahkan programmer dalam mempelajari macam method yang ada dalam Visual Basic .NET.

Event

Event bisa dianalogikan sebagai hasil dari sebuah tindakan oleh pengguna terhadap suatu komponen. Misal : jika pengguna melakukan klik kiri pada sebuah tombol, maka akan menimbulkan sebuah event *Click* dari button atau tombol tersebut.

Sebuah respon dari tindakan bisa menimbulkan beberapa event sekaligus. Misalnya jika kita menggerakkan mouse ke atas sebuah button, maka pada button tersebut bisa timbul event untuk *MouseMove* sekaligus *MouseOver*.

Appendix – Variabel dalam Visual Basic 2008

Variabel adalah penampung sementara dari sebuah tipe data tertentu di memori komputer. Setiap variabel di dalam Visual Basic .NET secara default harus dideklarasikan terlebih dulu secara eksplisit dengan diarahkan ke sebuah tipe data tertentu. Kecuali, jika set *Option Explicit* diberi nilai *Off*. Pemberian nilai tersebut bisa dilakukan di dalam program ataupun di dalam property project di bagian *Build*.

Aturan Penamaan

Penamaan variabel dalam Visual Basic .NET mempunyai beberapa aturan yang secara umum hampir sama dengan bahasa pemrograman yang lain, yaitu :

1. Variabel harus diawali dengan huruf atau underscore, mis :
   xBaru, xBaru1, _xBaru1 **à** *Benar*
   2xBaru, ?Baru **à** *Salah*
2. Variabel tidak boleh mengandung tanda baca atau simbol, mis :
   Baru_Kata, KataBaru **à** *Benar*
   Baru Kata, Kata?Baru **à** *Salah*
3. Jika diawali dengan tanda underscore (_), maka minimal harus mengandung sebuah huruf atau angka didalamnya, mis :
   _x, _1 **à** *Benar*
   _?, _< **à** *Salah*
4. Nama variabel tidak boleh sama dengan *keyword* di Visual Basic .NET , mis :
   DateBaru, StringBaru **à** *Benar*
   Date, String **à** *Salah*

5. Maksimal penamaan sebanyak 1023 karakter
6. Penamaan variabel tidak *case sensitive* (tidak membedakan antara huruf besar dan kecil), mis :

   VariabelSatu sama dengan penamaan variabel *variabelsatu*

Usahakan untuk memberi nama variabel secara konsisten dan mempunyai ciri yang sesuai dengan selera Anda. Hal ini agar memudahkan saat melakukan revisi program.

Tipe

Di Visual Basic .NET terdapat banyak sekali tipe data untuk penamaan variabel. Secara global, tipe data dibagi menjadi tiga bagian yaitu :

1. Numerik, terdapat dua macam yaitu :
    a. Integral

        Umumnya digunakan untuk representasi data dari angka bulat tanpa memperdulikan pecahan. Terdapat tiga jenis tipe data numerik integral secara garis besar, antara lain :

        i. Integer, jangkauan angkanya antara -2,147,483,648 sampai dengan 2,147,483,647. Sedangkan untuk tipe data *Int16* mempunyai jangkauan angka antara -32768 sampai dengan 32767. Untuk tipe data *Int64* mempunyai jangkauan angka antara -9,223,372,036,854,775,808

sampai
9,223,372,036,854,775,807.

ii. Short
Mempunyai jangkauan angka yang sama dengan tipe data *Int16*.

iii. Long
Mempunyai jangkauan angka yang sama dengan tipe data *Int64*

b. Non integral
Tipe ini digunakan untuk merepresentasikan data numerik yang memiliki pecahan atau angka di belakang koma. Terdapat tiga jenis yaitu :

i. Decimal
Jika digunakan untuk menampung bilangan yang tidak memiliki pecahan, maka jangkauan angkanya adalah plus minus 79,228,162,514,264,337,593,543, 950,335, sedangkan jika memiliki pecahan hingga 28 angka di

belakang koma, jangkauannya adalah plus minus 7.92281625142643375935439503 35.

ii. Single

Untuk jangkauan angka negatif antara -3.4028235E+38 hingga -1.401298E-45, sedangkan untuk jangkauan angka positif antara 1.401298E-45 hingga 3.4028235E+38

iii. Double

Untuk jangkauan angka negatif antara -1.79769313486231570E+308 sampai dengan -4.94065645841246544E-324, dan untuk jangkauan angka positif adalah 4.94065645841246544E-324 hingga 1.79769313486231570E+308

2. Karakter

Terdapat dua macam tipe data untuk jenis karakter yaitu :

a. Char

   Hanya mampu menampung satu huruf.

b. String

   Mampu menampung hingga 65535 karakter.

3. Miscelaneous

   Tipe data yang masuk dalam golongan ini adalah semua tipe data yang tidak termasuk ke dalam tipe numerik dan karakter. Beberapa tipe data dari golongan ini yang umum digunakan antara lain :

   a. Boolean

      Merupakan tipe data khusus yang hanya bisa menampung dua macam data yaitu *True* dan *False*

   b. Date

      Tipe data ini bisa menampung data tanggal dan jam, tanggal saja atau jam saja. Dalam pendeklarasiannya, biasanya diapit dengan tanda # #

4. Object

   Tipe data ini adalah bisa juga disebut sebagai tipe data bebas.

Konversi Tipe Data

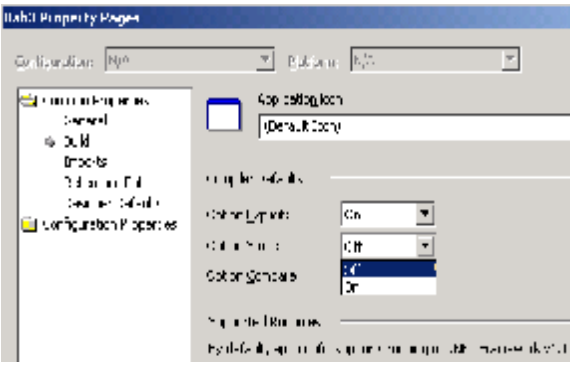
Sebuah variabel dengan tipe tertentu dapat dikonversi ke tipe yang lain dengan syarat bahwa tipe tersebut masih memungkinkan untuk dikonversi ke tipe yang akan dituju. Misal : jika kita memiliki sebuah variabel dengan tipe string dan berisikan nilai "123", maka secara logis kita masih dapat mengkonversi variabel tersebut ke tipe data numerik seperti integer. Tetapi jika kita memiliki sebuah variabel dengan tipe string dan berisikan nilai "ABC", maka secara logis, kita tidak bisa mengkonversi variabel tersebut ke tipe data numerik.

Konversi antar tipe variabel di dalam Visual Basic .NET terbagi menjadi dua macam yaitu :

1. Konversi secara implisit

   Merupakan konversi yang tidak memerlukan sebuah sintaks tertentu dalam implementasinya. Konversi secara implisit dapat dilakukan jika sebuah project sedang dalam kondisi Option Strict Off. Keadaan tersebut merupakan keadaan default yang

terdapat dalam sebuah project. Untuk mengubah agar tidak terjadi konversi variabel secara implisit, dapat diketikkan perintah Option Strict On di namespace level, atau dengan mengganti setting project melalui menu Project dan sub menu (nama project) Properties



Property project untuk Option Strict

Konversi secara implisit akan mempermudah proses dalam membuat program, dan membawa konsekuensi waktu eksekusi yang bertambah lamban. Tetapi, untuk sebuah project yang tidak terlalu besar kelambanan tersebut tidak akan terlalu dirasakan oleh pengguna.

Konversi secara eksplisit

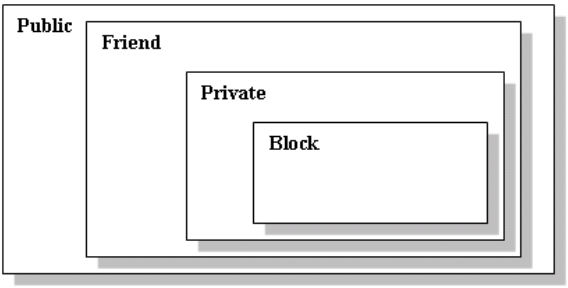
2. Konversi secara eksplisit merupakan konversi yang membutuhkan sintaks tertentu untuk mengubah sebuah tipe data ke tipe data yang lain, selama masih bisa dilakukan secara logis. Beberapa sintaks konversi yang seringkali digunakan adalah :
    a. CDate
       Mengkonversi dari tipe data string ke tipe data tanggal
    b. CInt, Val
       Mengkonversi dari tipe data string ke tipe data integer
    c. CStr, Str
       Mengkonversi dari tipe data lain (boolean, date, numerik) ke tipe string
    d. Chr, Chrw
       Mengkonversi kode angka ASCII ke karakter
    e. Asc, Ascw
       Kebalikan dari fungsi Chr dan Chrw
    f. Lcase, Ucase

Mengkonversi sebuah tipe data string ke huruf besar (Ucase) dan huruf kecil (Lcase)

g. DateSerial, TimeSerial
   Mengkonversi sebuah serial angka (berupa rangkaian bulan, tanggal dan tahun) ke tipe data date (DateSerial) atau waktu (TimeSerial)
h. DateValue, TimeValue
   Hampir sama dengan CDate
i. Day, Month, Year, Weekday
   Mengambil nilai tertentu dari tipe data date, misal : fungsi Day untuk mengambil tanggal dan seterusnya.

Jangkauan

Jangkauan sebuah variabel atau sering disebut *scope* adalah masa sebuah variabel tersebut bisa digunakan dalam sebuah solution di Visual Basic .NET. Di dalam Visual Basic .NET jangkauan sebuah variabel dapat dibagi menjadi beberapa level yaitu :



Skema level jangkauan variabel

1. Public

   Merupakan level tertinggi dalam jangkauan sebuah variabel. Sebuah variabel yang didefinisikan sebagai variabel *public* akan dapat diakses oleh semua project dalam sebuah solution. Deklarasi variabel dengan jangkauan public hanya dapat dilakukan di dalam sebuah

module atau di dalam *file level* dalam sebuah form atau di level namespace sebuah project.

Jika dideklarasikan dalam sebuah module, maka deklarasi variabel public bisa dilakukan di dalam sebuah procedure, sedangkan jika dideklarasikan dalam sebuah form, maka deklarasi variabel public harus diikuti dengan kata kunci *shared*, dan form tersebut harus dijadikan output sebagai sebuah *class library* agar dapat dijadikan *reference* oleh form lain yang akan menggunakan variabel tersebut.

2. Friend

   Jika sebuah variabel dideklarasikan dengan level ini, maka berarti variabel tersebut dapat diakses di dalam level program (module ataupun form).

3. Private

   Sebuah variabel dengan jangkauan *private* hanya dapat diakses dari konteks deklarasi yang bersangkutan, misalnya dalam satu procedure saja. Secara umum, jika sebuah variabel dideklarasikan di awal sebuah variabel

dan tanpa didahului dengan kata kunci *Private* (hanya didahului dengan kata *Dim*) dapat langsung diasumsikan sebagai sebuah variabel dengan jangkauan ini.

4. Block

   Merupakan jangkauan yang sama sekali baru di Visual Basic .NET dan tidak ada di dalam Visual Basic 6.0. Variabel dengan jangkauan ini merupakan variabel dengan jangkauan terpendek. Karena hanya dikenali di sebuah blok perintah, misalnya di dalam sebuah blok perintah *if..then..end if* atau di dalam sebuah blok perintah *for…next*

Array

Array (seringkali diterjemahkan sebagai senarai atau larik) merupakan set variabel yang dapat diakses dengan mengidentifikasi masing – masing item array dengan indeks angka tertentu. Di dalam Visual Basic .NET , indeks sebuah array selalu dimulai dari angka nol. Selain itu, sebuah array di dalam Visual Basic .NET merupakan sebuah obyek dari class array. Akibatnya, sebuah array dapat memiliki properti dan method sendiri yang diturunkan dari base class array.

Tipe data array sama dengan tipe data dari variabel, begitu juga dengan jangkauannya. Sebuah array juga mampu menampung data dengan tipe obyek ataupun control seperti button, textbox dan lainnya. Di dalam Visual Basic .NET , sebuah array dapat memanfaatkan method dari class array untuk menentukan ranking serta mengurutkan secara otomatis dari nilai array yang ada, bahkan bisa juga dimanfaatkan sebagai pointer.

Terdapat dua macam array berdasarkan dimensi yang dimilikinya, yaitu :

1. Array satu dimensi

   Merupakan array yang hanya memiliki satu urutan indeks

2. Array dua dimensi

   Merupakan array yang memiliki lebih dari satu urutan indeks.

Sebuah array memiliki batas atas dan batas bawah, yaitu batas minimal serta batas maksimal dalam satu deretan array itu sendiri. Batas bawah sebuah array dalam Visual Basic .NET selalu diasumsikan ke angka nol. Sedangkan batas maksimal sebuah array tergantung dari deklarasi array itu sendiri saat awal. Jika pada saat awal tidak terdapat deklarasi batas maksimal dari sebuah array, maka array tersebut akan menjadi sebuah array dinamis, yaitu array yang batas atasnya mengikuti pengisian data saat *run time*.

Appendix – Konsep Dasar Logika Pemrograman

Percabangan

If….then ….End if

Merupakan percabangan yang umumnya digunakan untuk percabangan dengan kondisi tunggal. Sintaks dari perintah ini adalah sebagai berikut :

```
If kondisi [ Then ]
   [ pernyataan benar  ]
[ ElseIf kondisi [ Then ]
   [ pernyataan benar] ]
[ Else
   [ pernyataan salah ] ]
End If
```

Jika perintah if hanya memerlukan satu pernyataan saja, maka tidak dibutuhkan penutup *end if.* Sedangkan untuk pernyataan if dalam satu baris, bisa digunakan juga perintah *IIF*

Select Case

Blok perintah *select case* merupakan blok perintah untuk percabangan majemuk dengan kondisi yang mempunyai tipe variabel yang sejenis.

Sintaks dari perintah ini adalah sebagai berikut :

```
Select [ Case ] kondisi
   [ Case daftar kondisi
      [ hasil ] ]
   [ Case Else
      [ hasil ] ]
End Select
```

Perulangan

For……Next

Perintah *For…Next* merupakan perintah yang ditujukan untuk perulangan suatu blok perintah dengan penghitung (counter) yang telah dibatasi pada awal baris blok perintah itu sendiri. Penghitung tersebut secara umum adalah sebuah variabel yang memiliki tipe numerik. Sintaks dari perintah ini adalah :

```
For counter  = awal To akhir [ Step langkah
]
   [ perintah ]
[ Exit For ]
   [ perintah ]
Next [ counter ]
```

Do While……Loop

Sintaks dari perintah ini adalah :

```
Do { While | Until } condition
  [ statements ]
[ Exit Do ]
  [ statements ]
Loop
atau
Do
  [ statements ]
[ Exit Do ]
```

```
  [ statements ]
Loop { While | Until } condition
```

Untuk mengaplikasikan perintah ini ke dalam program, disarankan untuk mendeklarasikan sebuah variabel yang berfungsi sebagai penghitung (counter) dan sebuah variabel sebagai pembatas (limit).

For Each ……. Next

Perulangan jenis ini merupakan perulangan yang dikhususkan untuk melakukan perulangan pada suatu *collection* baik dalam sebuah *structure* ataupun dalam sebuah komponen yang mengandung *collection* dari komponen yang lain seperti form, groupbox atau panel. Dengan menggunakan perulangan jenis ini, maka jumlah perulangan akan bergantung pada jumlah *item collection* yang ada. Sintaks dari perintah ini adalah :

```
For Each elemen In collection
   [ perintah ]
Next
```

Appendix – Prosedur dan Fungsi

Prosedur adalah blok perintah yang dapat dieksekusi dalam suatu program yang diawali dengan pernyataan deklarasi tertentu dan diakhiri dengan sebuah perintah *end* sebagai tanda akhir dari suatu prosedur. Tujuan dibuatnya sebuah prosedur adalah untuk memudahkan dan melakukan efisiensi dari suatu blok perintah yang dieksekusi berulang – ulang dalam suatu program (reusable).

Secara umum, sintaks dari sebuah prosedur adalah sebagai berikut :

```
[jangkauan] Sub nama prosedur [(parameter)]
   ' blok perintah
End Sub
```

Jangkauan sebuah prosedur ataupun fungsi sama dengan jangkuan yang berlaku dalam variabel (lihat lagi bab tentang variabel).

Secara garis besar, terdapat dua macam prosedur dalam Visual Basic .NET, yaitu :

1. Sub

Merupakan prosedur yang tidak mengembalikan nilai balik saat pemanggilan berlangsung.

2. Function (fungsi)

   Merupakan prosedur yang mengembalikan nilai balik saat pemanggilan berlangsung

Parameter merupakan data atau variabel yang diproses melalui sebuah prosedur. Terdapat tiga macam parameter di dalam Visual Basic .NET :

1. By Val

   Prosedur tidak bisa memodifikasi nilai asli dari parameter. Merupakan parameter default dalam Visual Basic .NET

2. By Ref

   Prosedur bisa memodifikasi nilai asli dari parameter

3. Exception

   Elemen yang dianggap bukan variabel, tidak akan bisa dimodifikasi meskipun didefinisikan sebagai *By Ref*

Sebuah parameter juga bisa didefinisikan sebagai parameter *optional* yaitu parameter yang bisa diisi ataupun tidak diisi saat pemanggilan prosedur berlangsung. Syarat dari pendefinisian tersebut adalah dengan menambahkan kata kunci *optional* mendeklarasikan nilai *default* dari parameter tersebut, jika parameter tidak diisi.

Pemanggilan sebuah fungsi (function) secara umum selalu diarahkan ke sebuah variabel sebagai penampung nilai balik dari fungsi tersebut. Karenanya, sebuah fungsi lebih sering digunakan untuk operasi – operasi matematika atau perhitungan lainnya.

Sintaks fungsi sama dengan sintaks sebuah prosedur, tetapi karena sebuah fungsi dapat memiliki nilai balik, maka pada saat deklarasi, juga biasa dideklarasikan tipe dari fungsi itu sendiri yang akan sekaligus menjadi tipe dari nilai balik yang dikembalikan.

Di dalam Visual Basic .NET terdapat istilah *handles* yang menyatakan bahwa suatu prosedur tertentu (termasuk method) menangani sebuah

event tertentu. Dengan adanya pernyataan *handles* maka bisa diasumsikan bahwa sebuah prosedur mampu menangani penggunaan beberapa control sekaligus (terutama yang sejenis).

Banyak programmer Visual Basic .NET yang berasal dari bahasa pemrograman Visual Basic 6.0 mengasumsikan teknik ini sebagai pengganti dari penggunaan control array yang populer digunakan di Visual Basic 6.0 tetapi sudah hilang saat Visual Basic .NET dikeluarkan.

Appendix – Konsep Form Majemuk

Pemakaian form majemuk berarti bahwa sebuah program atau aplikasi memiliki lebih dari satu form saat dijalankan. Terdapat dua macam teknik pemakaian form majemuk di semua bahasa pemrograman berbasis visual :

1. Single Document Interface
2. Multiple Document Interface

Penggunaan teknik form majemuk SDI, akan mengasumsikan bahwa tiap form akan berdiri sendiri tanpa struktur induk – anak (parent – child), sehingga programmer lebih bebas menentukan form mana yang akan diaktifkan.

Terdapat dua macam teknik dalam penggunaan SDI yaitu :

1. Form Modeless

   Mengijinkan pengguna untuk dapat berpindah antar form secara dinamis.
2. Form Modal

   Mengasumsikan bahwa form majemuk memiliki hirarki tertentu, sehingga form yang lain diasumsikan tidak bisa diproses sebelum form yang sedang aktif sudah selesai ditutup.

Berbeda dengan teknik SDI, teknik Multiple Document Interface atau MDI memiliki struktur induk – anak (parent – child) sehingga cara pembuatannya di dalam Visual Basic .NET sedikit berbeda dibandingkan dengan teknik SDI.

Teknik MDI akan menyebabkan semua form yang dianggap sebagai child akan selalu berada di dalam form induk atau form parent. Banyak sekali aplikasi yang menggunakan teknik MDI, misal : Microsoft Office. Dalam teknik MDI, memungkinkan pengguna untuk membuka sebuah form secara berulang kali. Selain itu, semua form child dapat diatur secara langsung melalui form parent misal : pengaturan cascade (bertumpuk) atau tile (berdampingan).

Sebuah aplikasi form majemuk dengan menggunakan teknik MDI, memiliki satu form parent dan minimal satu form sebagai child. Sebuah form parent ditandai dengan adanya nilai true pada property *IsMDIContainer*. Saat form parent ditutup, maka secara otomatis seluruh form child yang sedang terbuka ikut tertutup.

Appendix – Error Handling

Penanganan kesalahan di dalam Visual Basic .NET telah mengalami pembaruan diantaranya dengan munculnya control baru bernama *error provider* serta penanganan eksepsi yang baru dengan penggunaan class didalamnya. Error provider sendri merupakan sebuah control yang akan muncul secara dinamis sebagai sebuah icon dengan tooltip didalamnya (keterangan yang muncul jika mouse diarahkan diatas areanya) disamping control yang akan divalidasi. Control ini juga merupakan tipe control yang tidak langsung terlihat saat form dieksekusi. Selain itu, dalam penggunaannya, programmer bisa menggunakan satu error provider saja untuk beberapa control sekaligus dalam sebuah form yang aktif.

Field Validation

Teknik ini akan mengasumsikan penanganan kesalahan dilakukan di tiap aksi pengguna pada masing – masing control. Teknik ini sering dianggap merepotkan, terutama jika dalam sebuah form terdapat sejumlah control yang harus divalidasi dan memiliki jenis validasi yang serupa. Tetapi di lain pihak, pengguna cenderung lebih menyukai jenis ini karena kesalahan akan langsung ditunjukkan di tiap isian. Selain itu, kelemahan dari teknik ini adalah pengguna seringkali kesulitan untuk keluar dari sebuah form jika sudah melakukan kesalahan, dan juga pengguna melakukan *by pass* jika terjadi sebuah kesalahan.

Untuk menangani teknik ini, programmer bisa menggunakan method *Validating* dan *Validated* yang terdapat hampir di semua form yang mampu menerima inputan, seperti : textbox, checkbox ataupun radiobutton. Method validating akan melakukan pengecekan kesalahan,

sedangkan method validated digunakan untuk membersihkan pesan kesalahan.

Form Validation

Berbeda dengan field validation, teknik ini akan melakukan pengecekan kesalahan di akhir proses dari sebuah form. Umumnya pengecekan dilakukan di tombol terakhir yang menyatakan proses seperti tombol untuk simpan.

Bagi programmer, teknik ini akan mempermudah proses pembuatan karena mengumpulkan semua proses dalam satu tempat, tetapi bagi pengguna, seringkali merasa kecewa, sebab kesalahan baru akan ditunjukkan pada akhir proses saja.

Try……Catch…….Finally

Penggunaan perintah Try…Catch…Finally akan melakukan penanganan kesalahan dari blok perintah yang terdapat antara try dan catch. Dan jika terjadi suatu kesalahan, maka program tidak akan berhenti, tetapi akan mengeksekusi blok program yang terdapat setelah catch, sedangkan pernyataan yang terdapat setelah finally merupakan blok program yang akan selalu dieksekusi tanpa memperdulikan adanya kesalahan atau tidak.

Jenis kesalahan yang ditangkap oleh perintah catch disebut sebagai *exception.* Jika tak ada definisi tertentu tentang kesalahan yang ada, maka semua kesalahan yang mungkin terjadi akan berusaha ditangkap oleh try…catch. Terdapat beberapa macam jenis exception, dan yang perlu diingat bahwa exception sendiri merupakan sebuah class yang bisa diturunkan sifatnya oleh programmer untuk menciptakan tipe kesalahan yang jauh lebih spesifik sesuai dengan kebutuhan.

Appendix – Sekilas Tentang OOP

Mengapa Harus OOP ?

Bagi Anda para akademisi (mahasiswa utamanya), seringkali berargumen saat mempelajari konsep OOP (Object Oriented Programming) atau juga diistilahkan dalam bahasa Indonesia sebagai Pemrograman Beorientasi Obyek. Secara umum, hampir seluruh perguruan tinggi selalu mengajarkan konsep OOP dengan menggunakan bahasa pemrograman Java atau C++.

Tanpa menafikan kekuatan OOP dari Java maupun C++, salah satu kelemahan dalam proses pembelajaran OOP adalah pengajaran konsep OOP yang terlalu *high tech* alias di atas awan. Pengajaran OOP dengan melakukan pembelajaran konsep umum (umumnya dengan melakukan analogi terhadap benda lain seperti kendaraan, rumah atau mahluk hidup) memang terbukti sangat sulit diterapkan di dalam proses belajar mengajar di kelas. Akibatnya, kebanyakan dari para mahasiswa pun seakan sulit mencerna mengenai konsep OOP itu sendiri.

Salah satu pertanyaan yang sering terlontar dari para pemula adalah mengapa harus repot mempelajari OOP ? Padahal di dalam pemrograman berbasis visual hampir seluruh proses dilakukan dengan metode *drag and drop* ? Begitu pula dengan proses pengetikan listing yang dianggap berbasis struktural dibandingkan OOP. Bukankah merancang dengan konsep OOP jauh lebih rumit dan lama dibandingkan dengan teknik pemrograman biasa ?

Tentu saja hal tersebut tidak dapat dijelaskan dalam sebuah paragraf yang sederhana. Tetapi secara umum, contoh yang terdapat di dalam buku ini setidaknya mampu menjelaskan mengapa OOP sangat diperlukan, bahkan di era pemrograman visual sekalipun.

Jika Anda perhatikan bahwa di dalam pembuatan tiap form Visual Basic, selalu diawali dengan deklarasi *Public Class* yang berarti bahwa sesungguhnya Visual Basic adalah sebuah bahasa pemrograman yang mendukung implementasi konsep OOP sepenuhnya. Sayangnya, dengan

citra dari para programmer (terutama dari programmer *masa lalu*) yang menyatakan bahwa Visual Basic adalah bahasa pemrograman level *kiddies* dan tidak layak dipelajari karena dianggap tidak mendukung konsep OOP.

Sehingga jika ditelaah lebih lanjut, konsep OOP sesungguhnya wajib dipelajari oleh para programmer *masa kini*, baik yang mendalami bahasa pemrograman model visual maupun bahasa pemrograman model klasik. Karena hampir seluruh bahasa pemrograman modern saat ini dipastikan menggunakan konsep OOP didalamnya.

Visual Basic .NET sendiri sebagai bagian dari .NET Framework memang memudahkan para pemula untuk mempelajari konsep OOP. Tetapi bukan berarti bahwa mempelajari OOP adalah hal yang mudah, karena memang proses belajar dari sebuah konsep dasar tidak pernah mudah.

Buku ini sendiri memang tidak ditujukan untuk Anda yang secara lebih mendalam ingin mempelajari konsep OOP. Tetapi diharapkan

dengan contoh yang ada di dalam bab OOP, khususnya bagi Anda para pemula dapat lebih tertarik dan memahami mengapa OOP adalah sebuah kebutuhan untuk membuat aplikasi dengan level kompleksitas tinggi.

OOP Secara Visual ?

Di dalam pembahasan contoh aplikasi pada bab mengenai OOP, terdapa beberapa contoh aplikasi yang mencoba mengimplementasikan OOP secara visual. Implementasi OOP secara visual sesungguhnya sama dengan implementasi OOP biasa (lihat pada contoh pertama di bab OOP). Tetapi bagi para pemula khususnya, implementasi secara visual (berdasarkan pengalaman mengajar) umumnya lebih menarik untuk diaplikasikan.

Implementasi OOP secara visual dapat dilihat pada aplikasi yang menggunakan user control baik jenis *single* maupun *composite*. Karena di dalam pembuatan user control, segala sesuatunya (implementasi OOP) terlihat lebih mudah dibandingkan harus memulai segala sesuatunya dari awal (pembuatan *component*).

Implementasi OOP dengan media user control dapat dengan mudah dilacak kesalahannya dan yang terpenting adalah dapat mengakomodasi fungsi utama dari sebuah implementasi OOP di sebuah aplikasi yang dianggap kompleks. Selain

itu, diharapkan pula agar dengan contoh-contoh user control yang ada di alam bab OOP mampu merangsang Anda (khususnya pemula) untuk lebih banyak berkreasi dan mempelajari lebih lanjut mengenai konsep dan implementasi OOP.

Appendix – Sekilas Konsep XML

Apa itu XML ?

XML atau *Extensible Markup Language* merupakan bahasa *markup* yang mendeskripsikan data itu sendiri. Secara umum, XML pada dasarnya hanyalah sebuah file teks biasa dengan format tertentu. Format tertentu ? Ya, format yang sesungguhnya dapat secara bebas didefinisikan oleh tiap orang yang membuat file XML itu sendiri asalkan format itu dipenuhi secara konsisten dalam satu dokumen XML.

Karena file XML hanyalah sebuah file teks, maka file XML tidak mampu melakukan proses apapun, sehingga file XML membutuhkan alat bantu lain yang umumnya disebut sebagai *parser*.

Lalu, apa sesungguhnya kegunaan dari file XML itu sendiri ? Jika file XML hanyalah sebuah file teks, mengapa harus repot mempelajari XML ? Mengapa tidak menggunakan file teks biasa ?

File XML yang pada bulan Februari 1998 telah disetujui formatnya oleh W3C (World Wide Web Consortium), pada awal sejarahnya berasal dari SGML (Standard Generalized Markup

Language) yang telah menjadi sejak tahun 1985. Tujuan utama saat SGML dibentuk adalah untuk mempermudah pertukaran data antar program dan memiliki sifat *self describing*. Salah satu implementasi dari konsep SGML diantaranya adalah HTML yang mampu menjadi standar di penampilan informasi melalui internet.

Tetapi, pada pemikiran selanjutnya, HTML dianggap belum cukup mumpuni untuk melakukan pertukaran data. HTML lebih mengarah kepada cara untuk menampilkan data, bukan untuk *mendeskripsikan* data itu sendiri (self describing). Sehingga dari pemikiran ini muncullah ide pembentukan standar baru yaitu XML.

Lalu, apa keunggulan utama dari XML yang sesungguhnya ? Jawabannya adalah : **fleksibilitas** !! Kemampuan XML sebagai file teks yang dapat dibaca di semua editor teks di platform apapun, pemisahan XML dari program itu sendiri (sehingga perubahan file XML tidak akan mempengaruhi jalannya proses, tidak seperti pada HTML), dan dengan standar yang telah diakui oleh

W3C, maka XML dapat dibaca oleh semua browser yang tersedia dengan baik.

Dan, dengan kemampuan self describing yang dimilikinya, XML tidak hanya dimanfaatkan untuk kepentingan dalam browser, tetapi lebih ke arah pertukaran data itu sendiri. Baik pertukaran data antara satu database ke database yang lain, antar satu aplikasi ke aplikasi yang lain dan juga yang tak kalah penting adalah pemanfaatan XML dalam sebuah XML Web Service yang akan dijelaskan lebih lanjut di bab berikutnya.

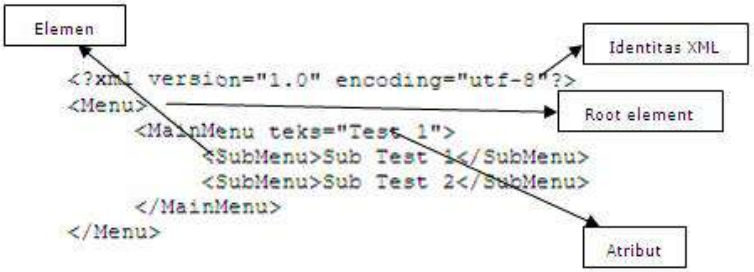
Menurut W3C, tujuan utama dari desain XML adalah :

1. XML harus dapat langsung digunakan melalui Internet.
2. XML harus mendukung penggunaan berbagai aplikasi
3. XML harus kompatibel dengan SGML.
4. Pembuatan program yang menggunakan XML harus mudah
5. Fitur piihan dalam XML harus seminimal mungkin, jika mungkin tidak boleh ada.

6. Dokumen XML harus jelas dan mudah dibaca oleh manusia.
7. Desain XML harus dapat dilakukan dengan sangat cepat
8. Desain XML harus formal dan konsisten
9. Proses pembuatan XML dokumen harus mudah
10. Tidak penting adanya markup XML yang rapi dan singkat (tidak seperti halnya HTML yang selalu singkat, markup dalam XML bisa saja panjang sesuai dengan kebutuhan pembuatnya).

XML Secara Aplikatif

Secara umum struktur XML dapat digambarkan sebagai berikut :



Identitas XML seringkali disebut juga sebagai *processing instruction.* Isi dari processing instruction adalah *version* yang merupakan rekomendasi dari W3C (hingga buku ini ditulis masih pada versi 1.0), dan *encoding* yang merupakan penanda bahwa model yang digunakan adalah huruf latin (UTF-8), untuk model lain dapat dilihat di situs W3C (www.w3c.org).

Markup dalam sebuah file XML dapat dianalogikan sebagai sebuah struktur kerangka tulang dalam tubuh manusia yang bertugas

sebagai penyangga utama. Sedangkan tag adalah bagian dari markup itu sendiri.

Secara sederhana, elemen merupakan frasa atau kata yang diapit oleh *start tag* (tag awal) yang berupa tanda **<.... >**dan *end tag* (tag akhir) yang berupa tanda **</....>** Jika sebuah elemen tidak memiliki isi data, maka end tag dapat langsung ditambahkan dalam start tag, misal : **<Menu />**.

Penamaan elemen secara umum memiliki aturan yang hampir sama dengan penamaan variabel dalam sebuah program, seperti case sensitive (membedakan antara huruf kecil dan huruf besar), tidak mengandung spasi dan juga simbol (tanda baca) kecuali tiga simbol yaitu tanda *hypercolon, underscore dan period* (-,_ dan .). Selain itu, nama elemen hanya boleh diawali dengan alfabet serta underscore.

Isi dari elemen tidak boleh mengandung beberapa karakter yang dianggap ilegal, dan jika ada maka harus digantikan dengan karakter subtitusinya, yaitu

Karakter Ilegal dalam XML	Karakter Subtitusi
<	<
>	>
'	'
&	&
"	"

Elemen sendiri mendeskripsikan data yang ada didalamnya. Dalam pembacaan XML sebagai Dataset, elemen juga seringkali dianalogikan seperti kolom atau field dalam sebuah tabel di database. Sebuah elemen juga dapat mengandung sub elemen lain. Pada penggambaran file XML sebagai tree, sebuah elemen diasumsikan

Sedangkan sebuah atribut secara fisik merupakan bagian dari elemen yang juga seringkali disebut sebagai *name value pair*. Name value pair berarti bahwa sebuah atribut terdiri dari nama atribut serta isi dari atribut itu sendiri. Sebuah atribut wajib memiliki isi atau value, meski isi dari atribut tersebut hanyalah string kosong (“”).

Sebuah elemen dapat terdiri dari banyak atribut, dengan syarat bahwa tiap atribut memiliki

nama yang *unique* serta memiliki aturan penamaan sama dengan penamaan elemen. Salah satu kelemahan atribut (terutama dalam aplikasi yang murni berbasis XML, bukan aplikasi database yang mengarahkan tabel ke dalam XML Schema) adalah kerancuan antara isi data dalam elemen dengan isi data dalam atribut. Sebuah file XML tidak wajib memiliki atribut didalamnya. Tetapi wajib memiliki minimal satu elemen didalamnya sebagai *root element*.

Dalam metode akses perbedaan utama antara elemen dan atribut adalah pada saat melakukan pencarian data. Sedangkan di banyak buku, pemilihan penggunaan elemen dan atribut seringkali berbeda. Sesuai dengan fungsinya, elemen lebih merepresentasikan node dalam tree, sedangkan atribut lebih sebagai isi atau aksesori dari elemen itu sendiri.

Appendix - Pengenalan GDI+

GDI+ ?

GDI+ (Graphics Drawing Interface) merupakan bagian .NET Framework yang menangani manipulasi grafis seperti grafik vektor, imaging ataupun tipografi. Secara umum, GDI+ mampu melakukan render grafis ke sebuah tampilan (baik desktop maupun web) secara dinamis. Kelebihan ini sering dimanfaatkan untuk melakukan generate gambar secara dinamis ataupun manipulasi bentuk – bentuk khusus.

Bagi Anda yang berkecimpung di dunia akademik, khususnya di teknik informatika, GDI+ dapat dimanfaatkan untuk pembelajaran bidang keilmuan pengolahan citra ataupun komputer grafika. Meski demikian GDI+ yang terdapat di dalam .NET Framework masih sering dianggap sebelah mata oleh para akademisi dalam kaitannya dengan implementasi konsep pengolahan citra.

Hal tersebut diakibatkan GDI+ dalam .NET Framework secara sederhana banyak bergantung pada kapabilitas *built in* class yang ada di dalam

.NET Framework. Tetapi sesungguhnya, jika dipelajari lebih dalam, GDI+ di dalam .NET Framework sepenuhnya dapat digunakan untuk pengolahan citra.

Implementasi GDI+

Di dalam .NET Framework, terdapat dua jenis citra digital yang diolah dengan GDI+ yakni bitmap dan vektor. Dalam contoh yang terdapat dalam buku ini, dapat terlihat jelas bahwa langkah umum dalam implementasi GDI+ adalah dengan mendeklarasikan obyek bertipe bitmap jika citra digital diambil dari sebuah file yang sudah *jadi*.

Sedangkan untuk menggambar suatu obyek citra digital dari awal, maka dapat digunakan obyek bertipe image. Dari obyek tersebut nantinya dapat dimanipulasi dengan berbagai jenis *built in function* dalam GDI+ seperti membuat garis (line), kurva hingga ke penulisan teks berbasis grafik.

Dan jika pengolahan citra digital telah selesai dilakukan, maka keluaran yang diharapkan dapat langsung disimpan dalam bentuk file atau secara temporer di sebuah *memory stream*. Teknik memory stream juga seringkali digunakan dalam pemrograman web, Anda dapat melihatnya pada pembuatan *captcha* yang saat ini umum

terlihat di proses registrasi situs-situs terkenal di dunia.

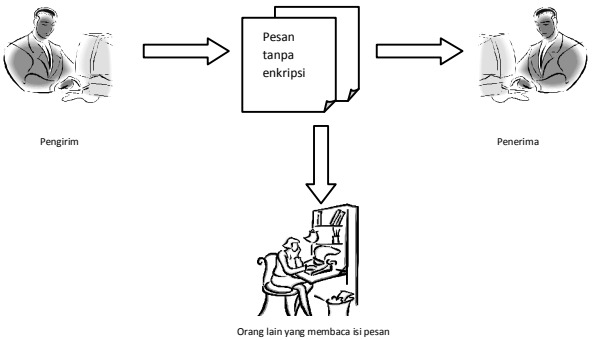
Appendix – Sekilas Kriptografi

Pengantar Kriptografi

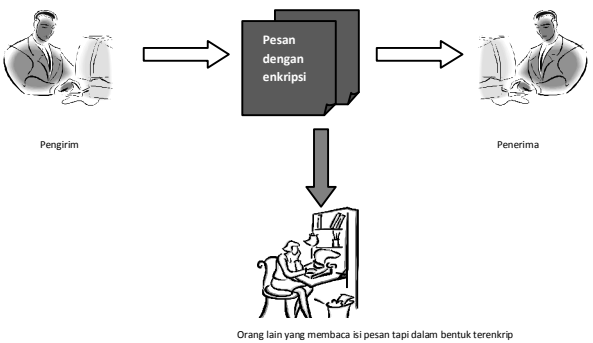
Jika Anda melakukan pengiriman surat melalui kantor pos, apakah bedanya jika pesan yang dikirim ditulis dalam sebuah kartu pos atau dalam selembar kertas yang dimasukkan ke dalam amplop tertutup ? Tentu saja jawabannya sederhana, tingkat kerahasiaannya. Pesan yang tertulis dalam selembar kartu pos akan bisa dibaca setiap orang yang memegang kartu pos tersebut, sedangkan pesan yang tersimpan dalam amplop hanya dapat dibaca jika amplop tersebut berhasil dibuka.

Tapi, apa yang terjadi jika amplop tersebut ternyata dibuka oleh orang yang sebenarnya tidak boleh membaca surat tersebut ? Maka kerahasiaan surat akan menjadi hilang, dan sia-sialah usaha Anda untuk menutupi pesan tersebut. Tetapi, jika isi dalam surat tersebut ternyata berisi tulisan yang berisi kode-kode tertentu yang hanya bisa dibaca oleh si penerima, apakah orang yang memaksa membuka amplop surat tersebut akan dapat membaca isi dari surat itu ? Tentu saja tidak

! Secara umum, proses melakukan pengkodean dari isi surat tersebut adalah proses enkripsi dalam kriptografi.



Skema pengiriman pesan tanpa enkripsi



Skema pengiriman pesan dengan enkripsi

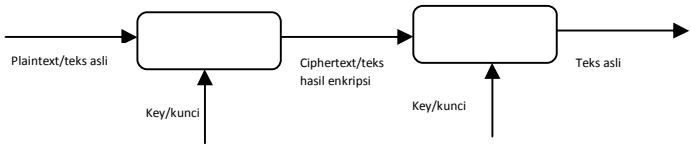
Istilah kriptografi (*cryptography*) berasal dari bahasa Yunani yaitu *kryptos* yang berarti tersembunyi dan *graphien* yang berarti menulis

atau jika diterjemahkan secara bebas berarti seni dan ilmu dalam menulis secara tersembunyi atau rahasia. Seringkali juga diartikan sebagai praktek penyembunyian data agar data atau pesan tersebut hanya bisa dibaca oleh orang yang berhak.

Jika sebuah pesan yang telah dikodekan berusaha untuk dipecah, maka proses tersebut disebut sebagai *kriptanalisis* (*cryptanalysis*), dan orang yang melakukannya disebut sebagai seorang *kriptanalis* (*cryptanalyst*).

Istilah kriptografi sendiri berbeda dengan istilah *kriptologi* (*cryptology*), karena kriptologi merupakan cabang ilmu matematika yang menggabungkan antara kriptografi dengan kriptanalisis. Berbeda lagi dengan istliah *kriptosistem* (*cryptosystem*) yang merupakan algoritma dengan segala kemungkinan teks asal (plaintext), teks hasil pengkodean (ciphertext) dan semua kemungkinan kunci yang ada.

Secara umum, sebuah proses dalam kriptografi dapat digambarkan dalam skema berikut ini :



Skema Umum Kriptografi

Penutup

Saat Anda telah selesai mempraktekkan seluruh contoh yang ada di dalam buku ini, maka pertanyaan selanjutnya adalah "What next ?". Apa yang harus dipelajari selanjutnya ?

Tentu saja untuk menjawab pertanyaan tersebut sangatlah relatif. Bagi Anda yang berkecimpung di dunia akademik, mungkin Anda membutuhkan beberapa implementasi tingkat lanjut secara teoritis seperti lanjutan GDI+, proses multi threading ataupun implementasi WMI (Windows Management Instrumentation) yang mampu mengakses perangkat keras. Sedangkan bagi Anda yang berada di bidang profesional atau sistem informasi, maka Anda mungkin lebih membutuhkan contoh-contoh aplikasi yang mengarah ke akses database hingga pembuatan XML Web Service.

Buku ini sejak awal memang dirancang untuk terbit dalam mode jilid majemuk, yang berarti bahwa akan terdapat jilid berikutnya setelah terbitnya buku ini. Bagi Anda yang tertarik untuk mendalami lebih jauh Visual Basic 2008,

maka selayaknya Anda juga memiliki jilid kedua dari buku ini yang nantinya akan membahas teknik pemrograman Visual Basic dalam melakukan akses database atau ADO .NET.

Dan bila Anda merasa kesulitan dalam mengimplementasikan contoh-contoh yang ada di dalam buku ini, maka Anda juga dapat menghubungi penulis untuk mendapatkan file asli dari contoh aplikasi yang ada di dalam buku ini. Anda dapat menghubungi penulis melalui email di soetamrizky@yahoo.com atau juga berkomunikasi via jejaring sosial di http://www.facebook.com/soetam.rizky dan juga bisa mendapatkan informasi terbaru penulis di http://blog.soetamrizky.info .