PEMROGRAMAN VISUAL DENGAN VB 2005

Diktat Mata Kuliah Pemrograman Visual

SOETAM RIZKY WICAKSONO

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

Untuk istri dan putri kecilku

Kata Pengantar

Alhamdullillah, puji syukur kepada Yang Maha Raja penguasa semesta, akhirnya buku ajar ini dapat terselesaikan, meski selaksa kendala menghadang di semua langkah. Tak lupa ucapan terima kasih sedalam-dalamnya untuk semua orang yang telah secara langsung dan tidak langsung membantu dalam penulisan buku ini untuk seluruh rekan "senasib" di proses sertifikasi VB .NET STIKOM Surabaya (MEL, TEG, RAG, CLA dan YUS), juga untuk Pak Sholiq dan Mbak Winarti yang tak pernah lelah untuk mengingatkan deadline pengumpulan buku. Tak lupa seluruh mahasiswa kelas sertifikasi STIKOM Surabaya yang telah memberi inspirasi selama pembuatan buku ini berlangsung. Terima kasih pula kepada rekan-rekan di LPPM Universitas Ma Chung yang telah membantu dalam proses pengiriman buku teks ini, Mas Irfan dan Bu Anna.

Buku ajar ini tidak dimaksudkan untuk mengganti sepenuhnya buku MOC (Microsoft Official Curriculum) Introduction to Visual Basic .NET, meski hampir seluruh bagian penting dari buku MOC terdapat dalam buku ini, dengan beberapa pengembangan serta pengurangan materi. Buku ini lebih dimaksudkan untuk memudahkan pemahaman para mahasiswa dalam menempuh mata kuliah Pemrograman Visual I. Hal ini didasarkan pada pengalaman proses belajar mengajar Pemrograman Visual I, saat mahasiswa cenderung merasa enggan untuk membaca MOC yang relatif tebal (dan juga berbahasa Inggris), sehingga buku MOC lebih terkesan sebagai sebuah aksesori belaka. Karenanya, diharapkan agar buku ini mampu menunjang proses belajar mengajar, baik bagi mahasiswa maupun bagi para dosen.

Di dalam buku ini pula, dilengkapi dengan studi kasus pengembangan di bagian akhir, yang dapat dimanfaatkan sebagai "uji kemampuan" para mahasiswa setelah melampaui materi yang telah diberikan dalam kelas. Meski begitu, tidak tertutup kemungkinan (terutama bagi para dosen) untuk mengembangkan lebih lanjut, baik materi ataupun studi kasus pengembangan sesuai dengan perkembangan teknologi dan situasi yang terjadi di dalam kelas. Sedangkan soal latihan beserta jawaban dapat menjadi tindakan kelas "learning by doing" bagi para dosen pengajar mata kuliah ini.

Semoga buku ajar ini benar – benar memberi manfaat bagi seluruh komponen yang terkait, meski banyak kekurangan yang masih terjadi, dan berharap agar saran dan kritik yang konstruktif dapat membantu untuk perbaikan di masa yang akan datang.

Selamat berkarya !!

Bhakti Persada lt.2, Malang

Maret 2009 / Rabiul Akhir 1430 H

Daftar Isi

Prakata

Saat belasan (bahkan puluhan) buku mengenai pemrograman Visual Basic beredar di pasaran, banyak pula variasi yang muncul di tiap pembahasan yang ada dalam tiap buku tersebut. Bahkan tidak sedikit buku yang telah melakukan pembahasan secara spesifik ke dalam suatu topik yang masuk dalam kategori *advanced*. Meski demikian tidak banyak buku yang melakukan pembahasan (khususnya bagi para pemula) dengan pendekatan *class activity learning*, sehingga sangat sulit untuk mengimplementasikan buku tersebut sebagai bahan acuan untuk kegiatan belajar mengajar di dalam kelas.

Buku ini secara umum disusun berdasarkan kurikulum resmi dari Microsoft atau lazim dikenal sebagai MOC (Microsoft Official Curriculum), tetapi dengan melakukan berbagai penyesuaian serta menerapkan adaptasi yang didapat dari hasil pengalaman saat mengajar di dalam kelas. Dengan melakukan pendekatan kegiatan belajar mengajar tersebut, dan juga dilengkapi dengan berbagai soal latihan yang langsung dapat dipraktekkan di dalam kelas maupun laboratorium, maka buku ini dapat menjadi acuan bagi para dosen untuk melakukan kegiatan belajar mengajar.

Selain itu, di dalam buku ini juga dilengkapi dengan berbagai tugas mandiri yang dapat diberikan kepada para mahasiswa agar dapat menjadi acuan bagi para dosen dalam mengukur tingkat pemahaman dari materi yang telah diberikan. Dengan penyusunan bab-bab yang telah diuji urutannya dalam penyajian di kelas, diharapkan buku ini selain dapat digunakan sebagai bahan ajar juga dapat digunakan sebagai bahan ajar mandiri.

Bab-bab yang terdapat dalam buku ini antara lain :

- **Bab *Pengantar* à didalamnya membahas tentang sejarah dan perkembangan dari Visual Basic .NET serta pemahaman mengenai .NET Framework**
- **Bab *Lingkungan Visual Basic .NET* à membahas mengenai konsep dasar pengenalan saat seorang programmer berhadapan dengan IDE (Integrated Development Environment) sebelum memulai tahapan pemrograman.**
- **Bab *Pengantar Pemrograman* à menyajikan tentang struktur dari proyek di dalam lingkungan Visual Basic .NET**

- **Bab *Form dan Komponen Dasar* à membahas mengenai penggunaan dasar form serta pengenalan komponen-komponen dasar yang hampir pasti digunakan di dalam tiap aplikasi, baik sederhana maupun kompleks.**
- **Bab *Logika Pemrograman Dasar* à didalamnya berisi tentang pengantar algoritma pemrograman dasar beserta penerapannya dalam sebuah pemrograman visual. Dimulai dari pengenalan mengenai variabel hingga ke bentuk perulangan dan percabangan.**
- **Bab *Prosedur dan Fungsi* à membahas tentang penggunaan prosedur dan fungsi di dalam pemrograman visual serta memperkenalkan penggunaan teknik penggunaan prosedur dan fungsi dengan teknik *handles*.**
- **Bab *Menu dan Form Majemuk* à berisi mengenai cara melakukan implementasi aplikasi yang didalamnya memiliki form lebih dari satu tetapi tetap terintegrasi.**
- **Bab *Penanganan Kesalahan* à didalamnya memperkenalkan proses penanganan kesalahan yang dapat mengurangi *human error* saat aplikasi telah digunakan oleh pengguna.**
- **Bab *Komponen Standard Lain* à membahas tentang komponen-komponen standard yang seringkali terlewatkan oleh programmer pemula, tetapi di sisi lain juga sering dibutuhkan saat telah melampaui level aplikasi yang kompleks.**
- **Bab *Kasus Pengembangan* à bab ini berisikan kasus-kasus pengembangan yang dapat langsung dipraktekkan saat kegiatan belajar mengajar berada di dalam laboratorium. Kasus-kasus yang ada merupakan aktifitas *learning by doing* yang diharapkan dapat membuka wawasan untuk dapat menerapkan teori yang telah didapat ke dalam sebuah aplikasi.**
- **Bab *Kasus Tugas Mandiri* à berisi tentang contoh-contoh kasus yang didalamnya sengaja tidak ditampilkan solusi untuk tiap kasus tersebut. Kasus dalam bab ini diharapkan dapat diselesaikan secara mandiri oleh mahasiswa sebagai tugas di luar kelas. Sedangkan bagi pembaca yang menginginkan untuk bahan ajar mandiri, diharapkan kasus yang ada dalam bab ini dapat merangsang logika pemrograman untuk dapat menjadi lebih baik.**

PENGANTAR

Tujuan :

- **Mengerti sekilas sejarah Visual Basic**
- **Memahami perbedaan antara .NET Framework dan Visual Basic .NET**

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.

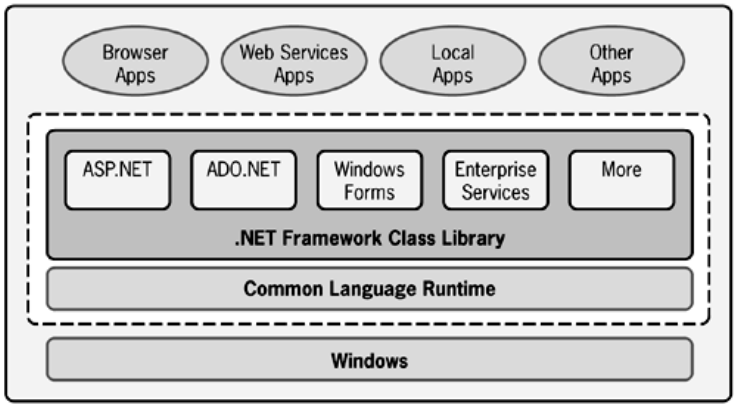
Sekilas .NET Framework

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.



Gambar 1.1. 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).

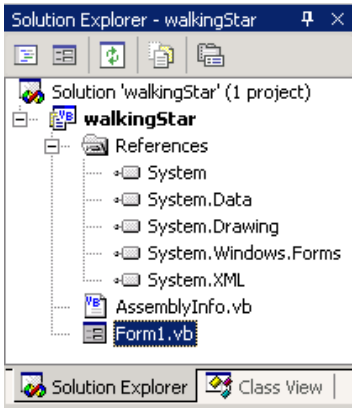
Lingkungan Visual Basic .NET

Tujuan :

- **Memahami IDE Visual Studio .NET**
- **Memahami penggunaan IDE Visual Studio .NET saat melakukan proses pengembangan aplikasi**

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

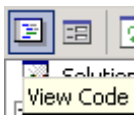


Gambar 2.1. Solution Explorer

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.

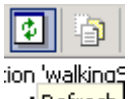
Solution explorer juga berguna untuk mengakses bagian – bagian dari sebuah project secara cepat dengan mengakses icon yang terdapat di bagian atas. Icon tersebut adalah :

1. 

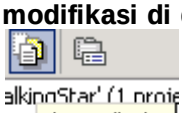
Berfungsi untuk menuju ke code editor sebuah form atau module

2. 

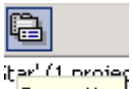
Berfungsi untuk menuju ke windows form designer

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

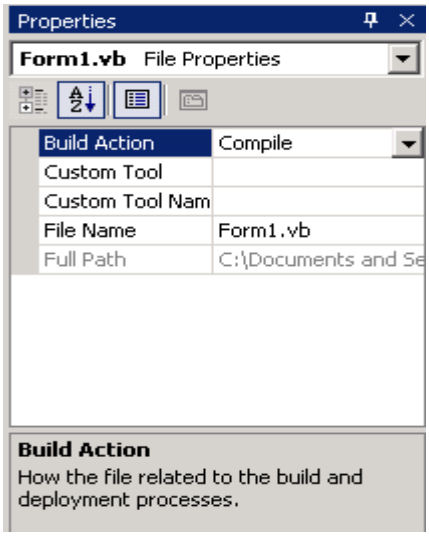
Properties window adalah window yang berisi properti dari sebuah control atau sebuah item di dalam project (module, form atau class) secara *design time*. Di dalam sebuah properties window terdapat icon – icon sebagai berikut :

1. 

Merupakan icon yang berfungsi untuk mengelompokkan properties berdasarkan kategori di kontrol tersebut.

2. 

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.



Gambar 2.2. Properties Window

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

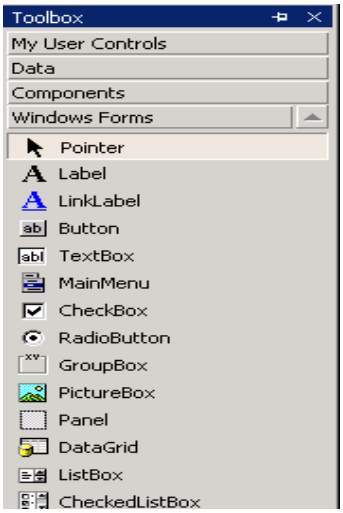
Windows Form Designer

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

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

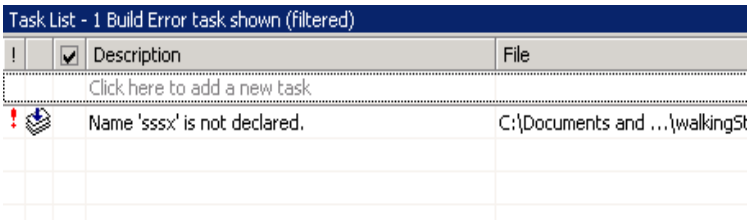


Gambar 2.3. Toolbox

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

Toolbox hanya akan menampilkan item saat Anda berada dalam form designer.

Task List



Gambar 2.4. Tasklist

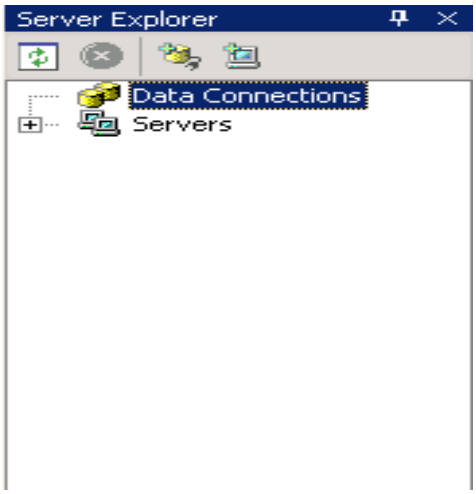
Tasklist merupakan salah satu window terpenting yang perlu diketahui oleh seorang programmer di lingkungan Visual Basic .NET. Secara detail, tasklist memiliki banyak fungsi yang sangat membantu dalam proses pemrograman. Tetapi fungsi terpenting sekaligus yang paling sering ditemui dari tasklist adalah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

Server Explorer

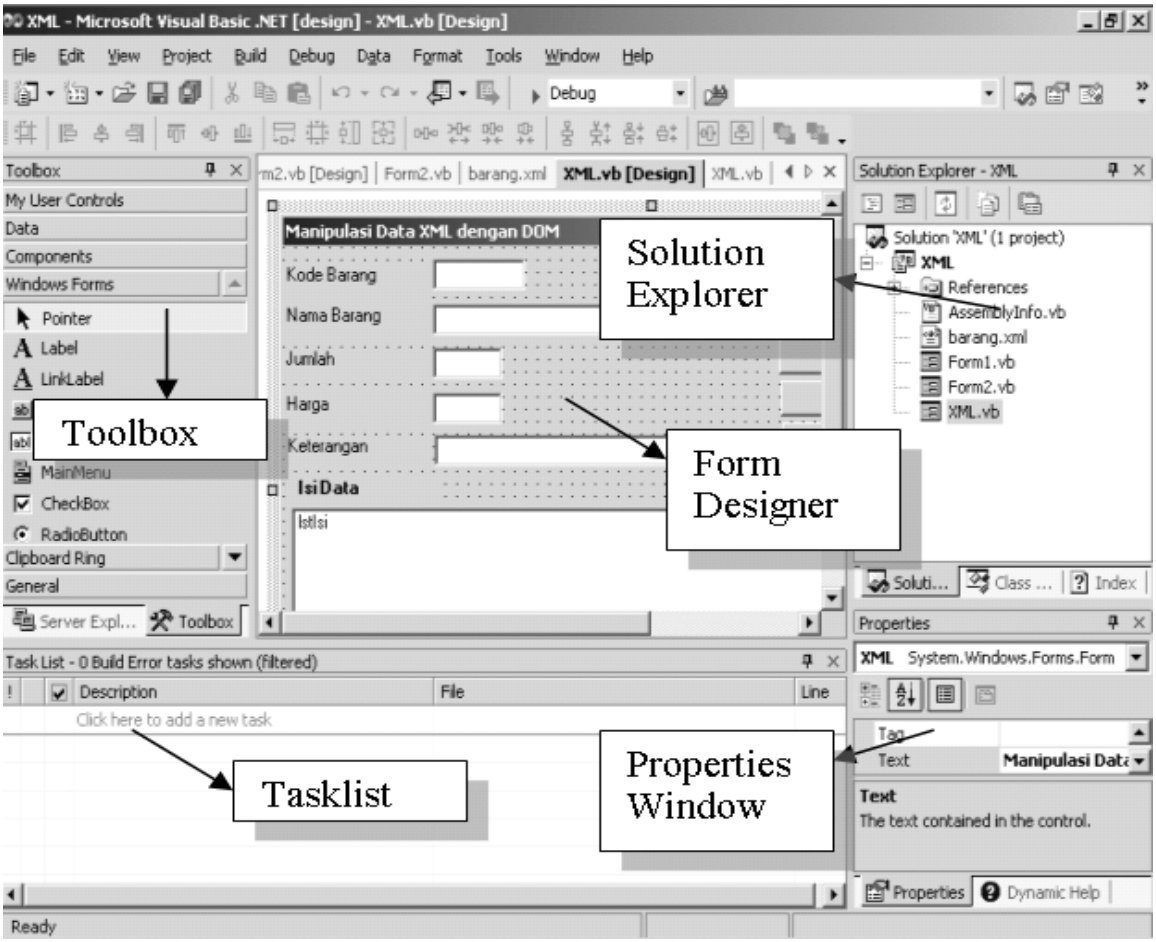
Server explorer adalah window khusus yang mampu melakukan koneksi dengan server database seperti Oracle, SQL Server atau database desktop seperti Access. Server explorer merupakan alat bantu baru di lingkungan Visual Basic .NET yang sangat membantu dalam pembuatan aplikasi database. Sebab, kemampuan 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.

Jika Anda belum melakukan instalasi database apapun (mis : Oracle, SQL Server, MS Access) di dalam komputer Anda, maka server explorer tidak berfungsi.



Gambar 2.5. Server Explorer



Gambar 2.6. Lingkungan Visual Basic .NET

PENGANTAR PEMROGRAMAN

Tujuan :

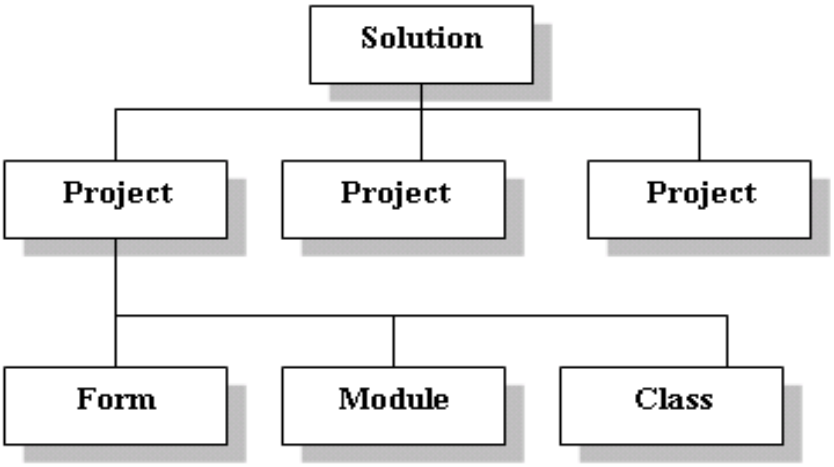
- **Memahami dasar pemrograman Visual Basic .NET**
- **Memahami dasar pembuatan serta persiapan pembuatan aplikasi dengan menggunakan Visual Basic .NET**

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.



Gambar 3.1 Struktur aplikasi Visual Basic .NET

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 – projectyang 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 hanya akan 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**

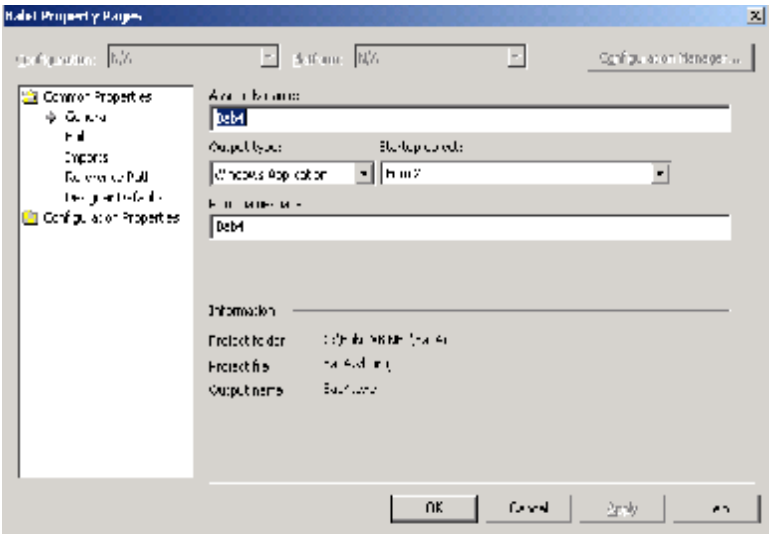
   Digunakan untuk membuat aplikasi instalasi yang digunakan untuk mendistribusikan aplikasi Visual Basic .NET ke komputer lain yang tidak terinstalasi Visual Basic .NET .

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.



Gambar 3.2 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.

2. **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*.

FORM DAN KOMPONEN DASAR

Tujuan :

- **Memahami jenis dan penggunaan property form serta komponen dasar yang diperlukan dalam pembuatan aplikasi dengan menggunakan Visual Basic .NET**
- **Mampu mengembangkan aplikasi sederhana dengan memanfaatkan komponen dasar yang terdapat dalam Visual Basic .NET**

Form adalah interface utama dari sebuah aplikasi windows standar di dalam Visual Basic .NET. Di dalam sebuah form terdapat beberapa property yang umum digunakan antara lain :

1. **Name**

   Menentukan nama dari form tersebut sebagai sebuah variabel dalam sebuah project.

   Keterangan

   Jika nama sebuah form yang menjadi startup diganti, maka property project harus diset ulang supaya startup form mengarah ulang ke form tersebut.
   Property project diset pada menu Project, lalu memilih sub menu properties. Setelah itu, pada kotak dialog yang muncul, ganti startup form ke nama form yang baru.

   Seluruh komponen selalu memiliki property

2. **AcceptButton**

   Menentukan tombol yang menjadi *default* saat pengguna melakukan penekanan tombol *Enter* di form

   Keterangan

   Property ini membutuhkan minimal satu komponen button di dalam sebuah form. Umumnya button yang diset sebagai sebuah AcceptButton akan bersifat seperti halnya tombol OK di berbagai kotak dialog di aplikasi MS Office, sehingga button tersebut akan bereaksi saat tombol Enter ditekan.

3. **BackColor**

   Menentukan warna dari sebuah form.

4. **CancelButton**

   Sama halnya dengan AcceptButton, tetapi yang ditangkap adalah penekanan tombol Esc.

5. **ControlBox**

   Jika diset nilainya menjadi *False* maka akan meniadakan icon Maximum, Minimum, Close serta icon yang ada di *title bar*.

- **Keterangan**

  Jika property ControlBox diset menjadi False, maka property MaximumBox, MinimumBox, dan Icon akan menjadi tidak berfungsi lagi

6. **Icon**

   Menentukan icon yang terletak di *title bar*.

7. **MaximizeBox**

   Menentukan apakah icon maximize yang terletak di *title bar* bisa diakses atau tidak.

8. **MaximumSize**

   Menentukan ukuran maksimal sebuah form dalam bentuk ukuran lebar dan tinggi.

9. **MdiChildren**

   Menentukan apakah form tersebut sebagai form MDI children dari sebuah MDI Parent.

- **Keterangan**

  Property ini hanya bisa diset jika terdapat minimal sebuah form lain yang telah diset menjadi MdiParent.

10. **MdiParent**

    Menentukan form sebagai sebuah MDI Parent form dari sebuah project.

11. **MinimumBox**

    Menentukan apakah icon minimize yang terletak di *title bar* bisa diakses atau tidak.

- **Keterangan**

  Jika property MinimumBox dan MaximumBox keduanya diset nilainya menjadi False maka kedua icon tersebut akan hilang.

12. **MinimumSize**

    Menentukan ukuran minimal sebuah form dalam bentuk ukuran lebar dan tinggi.

13. **Size**

    Menentukan ukuran dari sebuah form.

14. **StartPosition**

Menentukan posisi awal saat form dijalankan, terdapat 5 pilihan yaitu :

a. **CenterParent**

   Form diletakkan di tengah form *parent* (hanya berlaku jika sebuah project memiliki *multiple form* dengan mode MDI)

b. **CenterScreen**

   Form diletakkan tepat di tengah layar.

c. **Manual**

   Letak form ditentukan oleh property *Location*

d. **WindowsDefaultBounds**

   Letak form akan ditentukan oleh Windows

e. **WindowsDefaultLocation**

   Sama halnya dengan WindowsDefaultBounds, tetapi mempunyai dimensi yang sama dengan property *Size*

15. **Text**

Digunakan untuk mengganti teks yang terdapat di *titlebar* sebuah form.

16. **WindowState**

Property ini mengatur cara pemunculan form saat pertama kali dijalankan. Terdapat tiga macam pilihan yaitu :

a. **Normal**

   Form akan muncul dalam ukuran seperti yang ada di property *Size*

b. **Minimized**

   Form akan muncul dalam keadaan minimized

c. **Maximized**

   Form akan muncul dalam keadaan maximized (memenuhi layar).

Sedangkan untuk event yang sering dipakai adalah sebagai berikut :

1. **Load**

Event ini akan dipicu saat form pertama kali diload oleh project. Isi dari event ini umumnya adalah setting awal yang dimiliki oleh sebuah form.

2. **Closing**

Event ini akan dijalankan saat form ditutup dengan menggunakan perintah *Close* atau dengan penekanan icon *close* yang terletak di caption.

3. **Keypress**

   Event ini dipicu saat terjadi penekanan tombol jika form sedang dalam keadaan aktif. Event ini akan berjalan jika property *Keypreview* diset menjadi true.

Method form yang biasa digunakan dalam pemrograman yaitu :

1. **Show**

   Berfungsi untuk menampilkan form ke layar setelah form diinisialisasi. Method ini akan menampilkan form dalam mode *modeless* yang berarti form tidak menjadi prioritas dalam sebuah project.

2. **ShowDialog**

   Sama halnya dengan method show, tetapi akan menampilkan form dalam mode *modal* yang berarti bahwa form akan menjadi prioritas utama dalam sebuah project, dengan kata lain, form lain tidak bisa diakses sebelum form yang aktif ditutup terlebih dulu.

3. **Close**

   Menutup form sekaligus membebaskan sumber daya memori yang telah dipakai oleh form tersebut dalam memori.

4. **Dispose**

   Membebaskan form dari memori secara utuh.

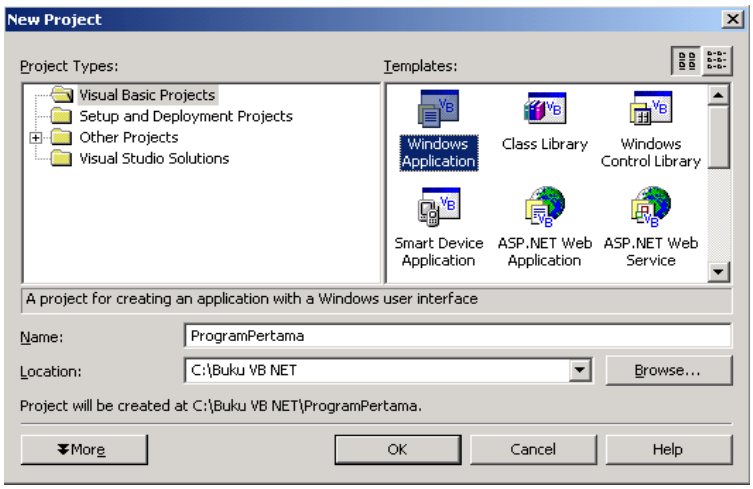
Soal latihan :

Set posisi form supaya berada tepat di tengah layar dan mempunyai ukuran 500 x 400 pixels. Tentukan juga agar form tidak bisa diganti ukurannya menjadi lebih kecil dari 300 x 400 pixels dan maksimal 640 x 480 pixels. Ganti teks pada caption menjadi *Form Pertama*. Lakukan semua setting property pada saat *design time*.

Jawab :

Untuk mengerjakan soal latihan tersebut, lakukan langkah – langkah berikut :

1. **Buka program Visual Basic .NET**
2. **Pada *Start Page* buat project baru untuk Windows Application, atau jika *Start Page* tidak ada, klik menu File dan pilih *New Project*.**

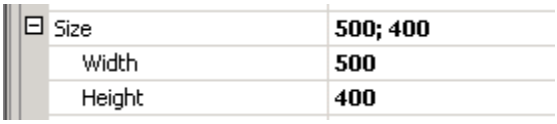


Gambar 4.1. Kotak dialog New Project

3. **Untuk melakukan setting property pada saat *design time*, maka property harus diset sebelum form dijalankan di property windows Set property pada jendela property sebagai berikut :**
4. **Form berada tepat di tengah layar**

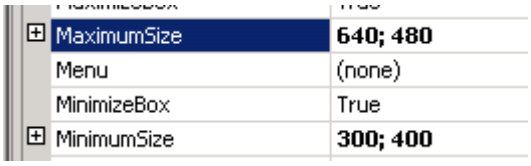
Gambar 4.2. Setting property Start Position

5. **Form berukuran 500 x 400 pixels**



Gambar 4.3. Setting property Size

6. **Ukuran form tidak boleh lebih kecil dari 300 x 400 pixels**
7. **Ukuran form maksimal 640x 480 pixels**



Gambar 4.4. Setting property MaximumSize dan MinimumSize

8. **Caption Form Pertama**

Gambar 4.5. Setting property Text

9. **Untuk melakukan tes terhadap form tersebut, jalankan program dengan menekan tombol F5 atau dengan melakukan klik pada icon Start yang terletak di toolbar. Kemudian lakukan cek dengan mengubah ukuran form melebihi ukuran maksimal dan mengurangi ukurannya menjadi lebih kecil dari ukuran minimal.**

Gambar 4.6. Icon Start

Soal latihan :

Buat sebuah form seperti pada latihan soal sebelumnya tetapi setting property dilakukan saat *run time*

Jawab :

Untuk menjawab soal latihan tersebut lakukan langkah – langkah berikut :

1. **Buat sebuah solution dan project baru di Visual Basic .NET**
2. **Pada *Windows Form Designer* lakukan klik ganda untuk menuju ke *Code Editor* atau dengan melakukan klik kanan dan memilih sub menu *View Code* selanjutnya memilih event *Load***

Gambar 4.7. Menu shortcut View Code

3. **Setelah itu, pada Code Editor ketikkan listing berikut :**

```
Private Sub Form1_Load(…… )
    With Me
        .CenterToScreen()
        .Size = New Size(500, 400)
        .MinimumSize = New Size(300, 400)
        .MaximumSize = New Size(640, 480)
        .Text = "Form Pertama"
    End With
End Sub
```

4. **Lalu jalankan program tersebut dengan menekan tombol F5**

Dalam keadaan default, saat Anda melakukan eksekusi sebuah aplikasi, maka Visual Basic .NET akan sekaligus menyimpan dan mengkompilasi solution. Hasil kompilasi berupa file dengan ekstensi *.exe* yang tersimpan di sub folder *bin*

Label

Label merupakan komponen yang seringkali digunakan hanya sebagai representasi dari sebuah teks yang memberikan keterangan. Secara umum, sebuah label adalah pelengkap di dalam sebuah aplikasi. Jarang sekali terdapat sebuah aplikasi yang memanfaatkan sebuah label hingga ke level event, karena interaksi pengguna lebih banyak diarahkan ke komponen yang lain misal : textbox.

Property yang seringkali digunakan untuk komponen ini adalah property *Text* yang mempunyai fungsi untuk mengganti tampilan teks yang akan muncul di label.

Textbox

Textbox adalah komponen yang mampu menerima inputan dari pengguna sebagai sebuah teks. Komponen ini seringkali digunakan sebagai inputan dasar dari sebuah aplikasi. Sebuah textbox mampu menampung maksimal 2048 karakter, dan jika property *multiline* diset nilainya menjadi true, maka textbox bisa menampung hingga 32 Kb teks.

Property yang seringkali digunakan untuk komponen ini antara lain :

1. **Name**

   Sama halnya dengan property *Name* di komponen yang lain.

2. **AcceptReturn**

   Menentukan apakah tombol *Enter* dapat berfungsi saat textbox dalam mode *MultiLine*

3. **AcceptTab**

   Menentukan apakah tombol *Tab* dapat berfungsi saat textbox dalam mode *MultiLine*

4. **CharacterCasing**

   Property yang dapat menentukan mode huruf yang diinputkan dalam textbox. Terdapat tiga pilihan yaitu :

a. **Normal**

   Huruf akan tampil apa adanya

b. **Upper**

   Semua huruf yang diinput akan dikonversi menjadi huruf besar

c. **Lower**

   Semua huruf yang diinput akan dikonversi menjadi huruf kecil

5. **Font**

   Menentukan jenis dan ukuran font di dalam textbox

6. **MaxLength**

   Menentukan jumlah karakter maksimal yang bisa diinput dalam sebuah textbox

7. **MultiLine**

8. **Jika diberi nilai true akan menjadikan textbox dalam mode *MultiLine* yaitu mampu menampung lebih dari satu baris.**

9. **PasswordChar**

10. **Memberi setting karakter khusus yang muncul saat textbox diinput, misal : dalam inputan password.**

11. **Text**

    Merupakan property default dalam textbox yang menyatakan nilai yang ada dalam textbox

12. **TextAlign**

    Menentukan *alignment* atau perataan yang ada dalam textbox, ada tiga pilihan dalam property ini yaitu :

    a. **Left : rata kiri**

    b. **Right : rata kanan**

    c. **Center : rata tengah**

    Method yang sering digunakan dalam komponen ini adalah :

1. **AppendText**

   Method ini digunakan untuk menambah karakter ataupun kata di dalam textbox. Berbeda dengan property *text* yang mengganti keseluruhan karakter, method ini lebih dikhususkan untuk menambahkan isi dari property *text*.

2. **Clear**

   Method ini dipakai untuk menghilangkan seluruh isi property *text*.

3. **Copy**

   Digunakan untuk melakukan duplikasi atau proses copy isi dari textbox ke dalam *clipboard* untuk diproses lebih lanjut di aplikasi lain atau di dalam form itu sendiri.

4. **Cut**

   Digunakan untuk melakukan pemindahan atau proses *cut* isi dari textbox ke dalam *clipboard* untuk diproses lebih lanjut di aplikasi lain atau di dalam form itu sendiri.

5. **Paste**

   Method ini hanya bisa digunakan jika *clipboard* telah terisi, baik dari hasil proses *copy* ataupun dari proses *paste*.

6. **Focus**

   Method ini digunakan untuk mengaktifkan textbox atau mengambil fokus dari aplikasi ke textbox tersebut. Di Visual Basic 6.0 method ini sama dengan method *setfocus*.

7. **SelectAll**

   Digunakan untuk melakukan *selection* atau pemilihan semua teks yang ada dalam textbox

8. **SelectNextControl**

   Method ini akan memindahkan fokus ke komponen lain yang nilai property *tabindex*-nya lebih besar 1 kali dari nilai property *tabindex* textbox tersebut.

Event yang umum dipakai adalah sebagai berikut :

1. **GotFocus**

   Event ini akan dipicu saat textbox menerima fokus atau pada saat user mengaktifkan textbox

2. **LostFocus**

   Kebalikan dari event GotFocus, event ini akan dijalankan saat textbox telah kehilangan fokus

3. **TextChanged**

   Event ini akan dipicu setiap kali isi dari textbox berubah, dengan kata lain, saat textbox dalam keadaan aktif dan pengguna mengetikkan sebuah karakter, maka event ini akan dijalankan.

4. **Validated**

   Event validated dijalankan sebelum event *LostFocus*. Event ini akan dipicu jika semua kondisi yang ada dalam event *Validating* telah berhasil dilampaui. Umumnya event ini dipakai untuk membersihkan pesan error yang sebelumnya dijalankan di event *Validating*.

5. **Validating**

   Event ini seringkali dipakai untuk mengecek kesalahan pemasukan data yang terjadi dalam sebuah textbox. Saat pengguna akan memindahkan fokus dari sebuah textbox, maka yang dipicu pertama kali adalah event Validating, kemudian event Validated, baru yang terakhir adalah event LostFocus.

Button

Di dalam Visual Basic 6.0, komponen ini dinamakan sebagai *command button*, sedangkan di Visual Basic .NET diganti namanya menjadi button. Fungsi utamanya tetap sama, yaitu sebagai tombol yang menerima event click dari pengguna.

Property yang sering dipakai untuk button adalah :

1. **FlatStyle**

   Property ini akan mengganti tampilan standard button jika nilainya diganti menjadi flat

2. **Text**

   Digunakan untuk mengganti nilai caption di button. Di Visual Basic 6.0 property ini setara dengan property *caption*. Untuk button, property text juga bisa menjadi *shortcut*

Method – method berikut umum digunakan untuk button :

1. **Focus (lihat di sub bab textbox)**
2. **SelectNextControl (lihat di sub bab textbox)**

Event yang sering dipakai untuk button adalah :

1. **Click**

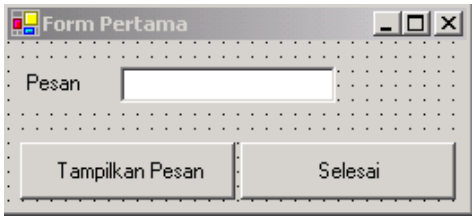
   Merupakan event default dari sebuah button. Event ini akan dipicu jika pengguna melakukan klik pada button

2. **MouseMove**

   Merupakan event yang terpicu dari pergerakan kursor mouse yang melewati area di button.

Soal latihan :

Buat form dengan layout seperti pada gambar berikut :



Gambar 4.8. Layout form latihan soal

Kemudian ketikkan listing program agar jika button *Tampilkan Pesan* diklik akan menampilkan pesan yang berisikan teks yang telah diketikkan di textbox. Sedangkan jika diklik button *Selesai* akan menutup form dan mengakhiri program.

Jawab :

1. **Buat solution dan project baru**
2. **Pada form set property sebagai berikut :**

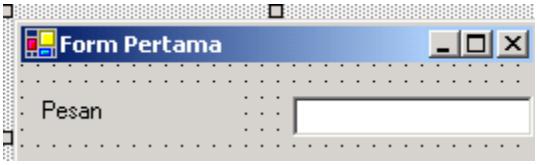
Size	264; 120
SizeGripStyle	Auto
SnapToGrid	True
StartPosition	CenterScreen
Tag	
Text	Form Pertama

Gambar 4.9. Property Form

3. **Drag label pada form, set property *text* menjadi *Pesan* sehingga layout form menjadi seperti pada gambar berikut :**

Gambar 4.10 Layout form dengan label

4. **Lalu drag sebuah textbox dari toolbox, ganti nilai property *text* menjadi kosong.**



Gambar 4.11 Layout form dengan label dan texbox

5. **Selanjutnya, drag dua buah button sehingga posisi kedua button sama dengan gambar di layout soal. Lalu, set property kedua button tersebut sebagai berikut:**

Size	120; 32
TabIndex	2
TabStop	True
Tag	
Text	**Tampilkan Pesan**

Gambar 4.12 Setting property button tampilkan pesan

Size	120; 32
TabIndex	3
TabStop	True
Tag	
Text	**Selesai**

Gambar 4.13 Setting property button selesai

6. **Untuk mengakses kode program pada button *Tampilkan Pesan*, lakukan klik ganda pada button tersebut di form designer, lalu ketikkan listing program berikut :**

```
Private Sub Button1_Click(ByVal ……
  MessageBox.Show(TextBox1.Text, _
      "Pesan pertama", MessageBoxButtons.OK, _
      MessageBoxIcon.Information)
End Sub
```

þ Keterangan

Perintah MessageBox merupakan perintah baru di Visual Basic .NET yang berfungsi sama dengan perintah MsgBox di Visual Basic 6.0. Sintaks dari perintah MessageBox adalah sebagai berikut :
MessageBox.Show(*pesan, judul, tombol yang ditampilkan, icon, nomor button yang menjadi default aktif, option tambahan)*

7. Selanjutnya untuk kode program pada button *Selesai*, lakukan klik ganda pada button tersebut di form designer, lalu ketikkan listing program berikut ini :

```
Private Sub Button2_Click(…….
    Close()
End Sub
```

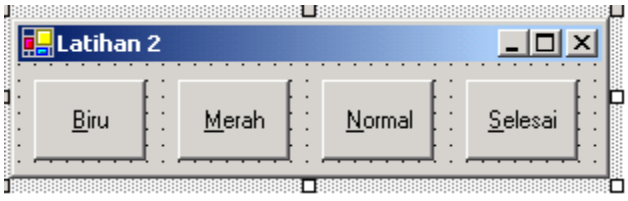
þ Keterangan

Method *Close* digunakan untuk menutup sebuah form sekaligus membebaskannya dari memori.

8. Setelah selesai, tekan F5 untuk menjalankan program.

Soal latihan :

Buat sebuah form, dengan layout seperti pada gambar berikut :

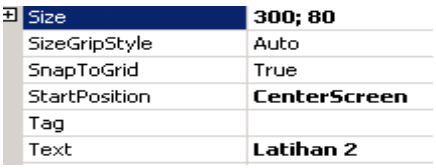


Gambar 4.14 Layout form soal

Kemudian ketikkan listing program untuk masing – masing button yang ada agar jika button tersebut diklik oleh pengguna akan mengganti warna latar belakang form sesuai dengan teks yang tertera di button tersebut.

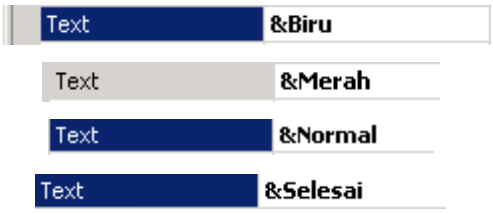
Jawab :

1. **Buat sebuah solution dan project baru**
2. **Pada form, set property seperti pada gambar berikut :**



Gambar 4.15 Property Form

3. **Drag empat buah button ke form sehingga layout form akan seperti pada soal**
4. **Set masing – masing property button sebagai berikut :**



Gambar 4.16 Property button

þ Keterangan

Pada property *text* di masing – masing button terdapat tanda *ampersand* (tanda &) yang mengawali tiap kata. Tanda tersebut merupakan penanda untuk shortcut key pada masing – masing button. Artinya bahwa jika tanda ampersand terletak sebelum huruf *B* pada button *Biru*, maka button tersebut bisa diakses dengan menggunakan kombinasi tombol Alt + B.

5. **Pada button *Biru* klik ganda, kemudian ketikkan listing berikut :**

```
BackColor = Color.Blue
```

þ Keterangan

Color pada Visual Basic .NET berbeda dengan di Visual Basic 6.0. Pada Visual Basic .NET, color merupakan sebuah *collection* dengan pilihan berbagai warna pallete, serta dapat juga diambil dari kombinasi ARGB (alpha (transparansi), red, green, blue (sebagai warna dasar). Selain itu , juga bisa diambil dari nama biasa serta nama themes Windows mis : ActiveControl, Control dan lainnya.

6. **Pada button *Merah* klik ganda, kemudian ketikkan listing sebagai berikut :**

```
BackColor = Color.Red
```

7. **Pada button *Normal* klik ganda, kemudian ketikkan listing berikut :**

```
BackColor = Color.Empty
```

þ Keterangan

Penggunaan kata kunci empty pada color collection akan mengembalikan nilai warna ke warna default dari Windows.

8. **Pada button *Selesai* klik ganda, kemudian ketikkan listing seperti berikut :**

```
Close()
```

9. **Jika selesai, tekan tombol F5 untuk menjalankan program**

Soal latihan :

Buat sebuah form, dengan layout seperti pada gambar berikut :

Gambar 4.17 Layout form soal

Kemudian ketikkan listing program untuk program tersebut dengan kondisi sebagai berikut :

- **Untuk tombol *Gabung* akan menggabungkan dua isi textbox yang ada ke dalam textbox *Hasil***
- **Tombol *Hitung Huruf* akan menghitung jumlah huruf dari isi textbox pertama dan kedua**
- **Tombol *Balik Pesan 1* akan membalik isi dari textbox pertama dan menampilkan hasilnya ke textbox *Hasil***

Jawab :

Pada soal tersebut, khusus untuk textbox yang terakhir, set property *Multiline* textbox menjadi *True*, agar ukuran textbox dapat memanjang dan mampu menampung teks dengan penggantian baris

Jika property Multiline, diset ulang menjadi false, maka ukuran tinggi textbox.akan kembali menjadi default.

Setelah membuat layout form seperti pada soal, ketikkan listing program berikut :

1. **Untuk tombol *Selesai*, ketikkan listing berikut :**

```
Close
```

2. **Pada tombol *Gabung* ketikkan listing berikut :**

```
TextBox3.Clear()
TextBox3.Text = TextBox1.Text
TextBox3.AppendText(vbCrLf & TextBox2.Text)
```

þ Keterangan

VbCrLf adalah singkatan dari Visual Basic Carriage Return Line Feed, yang berfungsi untuk menyisipkan penggantian baris.

3. **Untuk membalik kata pada tombol *Balik Pesan 1* ketikkan program berikut :**

```
TextBox3.Clear()
TextBox3.Text =  StrReverse(TextBox1.Text)
```

þ Keterangan

StrReverse adalah fungsi yang digunakan untuk membalik susunan huruf dari sebuah teks.

4. Pada tombol *Hitung Huruf* ketikkan listing berikut :

```
TextBox3.Text = _
    TextBox1.TextLength() + TextBox2.TextLength()
```

þ Keterangan

Dengan menggunakan property *TextLength*, maka kita bisa mengetahui berapa jumlah huruf yang terdapat dalam suatu teks yang berada di dalam sebuah textbox.

LOGIKA PEMROGRAMAN DASAR

Tujuan :

- **Mampu mengaplikasikan logika pemrograman dasar ke dalam aplikasi dengan menggunakan komponen yang terdapat dalam Visual Basic .NET**
- **Memahami penggunaan dan mampu mengembangkan aplikasi menggunakan komponen standard dalam Visual Basic .NET**

Variabel

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*
   Jika diawali dengan tanda underscore (_), maka minimal harus mengandung sebuah huruf atau angka didalamnya, mis :
   _x, _1 à *Benar*
   _?, _< à *Salah*
3. **Nama variabel tidak boleh sama dengan *keyword* di Visual Basic .NET , mis :**
   DateBaru, StringBaru à *Benar*
   Date, String à *Salah*
4. **Maksimal penamaan sebanyak 1023 karakter**
5. **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.

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 :

   b. **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.

2. **Short**

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

3. **Long**

   Mempunyai jangkauan angka yang sama dengan tipe data *Int64*

4. **Non integral**

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

5. **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 7.9228162514264337593543950335.

6. **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

7. **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

8. **Karakter**

   Terdapat dua macam tipe data untuk jenis karakter yaitu :

   a. **Char**

      Hanya mampu menampung satu huruf.

   b. **String**

      Mampu menampung hingga 65535 karakter.

9. **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**

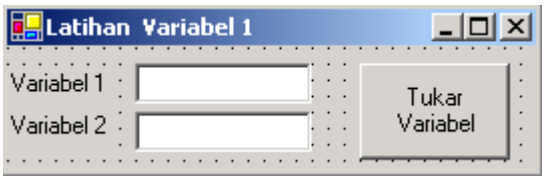
      Tipe data ini bisa menampung data tanggal dan jam, tanggal saja atau jam saja. Dalam pendeklarasiannya, biasanya diapit dengan tanda # #

   c. **Object**

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

Soal latihan :

Buat form dengan layout seperti pada gambar berikut :



Gambar 5.1. Soal latihan

Dengan kondisi, jika tombol *Tukar Variabel* ditekan, maka isi dari kedua textbox akan bertukar tempat. Gunakan variabel dengan tipe string untuk memenuhi syarat tersebut.

Jawab :

1. **Buat sebuah project baru**
2. **Pada form, tempatkan dua buah textbox, dua buah label dan sebuah button dengan layout seperti pada gambar.**
3. **Untuk kedua buah textbox, set property *Text* menjadi kosong**
4. **Pada button, ketikkan listing berikut :**

```
Dim var1 As String = TextBox1.Text
Dim var2 As String
var2 = TextBox2.Text
TextBox1.Text = var2
TextBox2.Text = var1
```

þ Keterangan

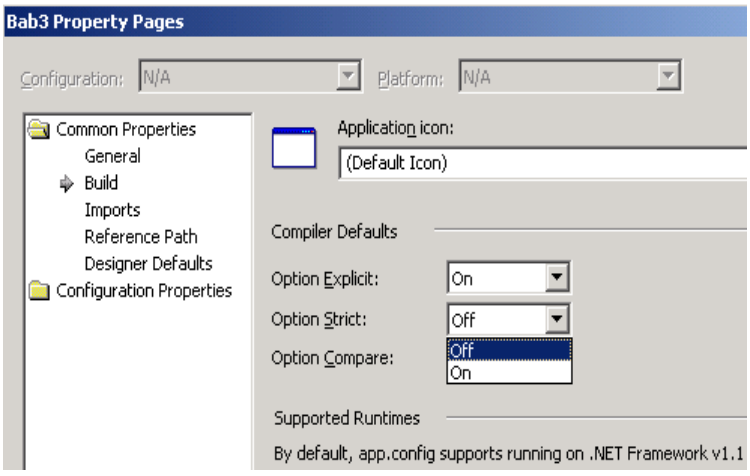
Pada jawaban soal latihan, terdapat dua macam cara deklarasi variabel. Cara pertama adalah dengan mendeklarasikan variabel sekaligus menempatkan nilai ke dalam variabel tersebut dalam baris yang sama. Sedangkan cara yang kedua adalah dengan mendeklarasikan variabel terlebih dulu, dan selanjutnya adalah mengisi variabel dengan nilai tertentu pada baris pernyataan tersendiri.
Cara pertama sangat disarankan untuk digunakan, terutama jika kita sudah mengetahui nilai dari sebuah variabel tersebut secara pasti.

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*

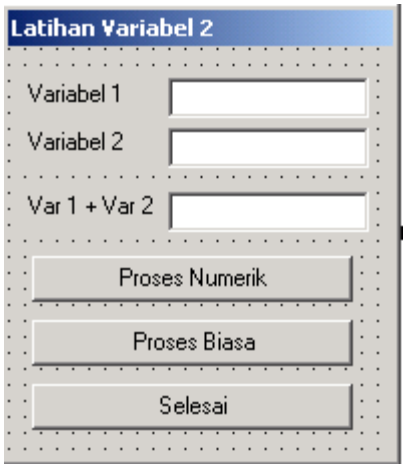


Gambar 5.2. Property project untuk Option Strict

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.

Soal latihan :

Buat form dengan layout seperti pada gambar, yang akan menampilkan hasil dari penambahan antara variabel 1 dan variabel 2 yang terdapat dalam textbox. Perlu diketahui bahwa property text dari sebuah textbox selalu bertipe string, dan aturan konversi secara implisit yang akan mengkonversi sesuai tipe dari isi variabel pertama dan kedua.



Gambar 5.3. Form untuk pembuktian konversi secara implisit

Jawab :

Pada tombol *Proses Numerik* ketikkan listing berikut :

```
Dim a, b As Integer
a = TextBox1.Text
b = TextBox2.Text
TextBox3.Text = a + b
```

Pada tombol *Proses Biasa* ketikkan listing berikut :

```
TextBox3.Text = TextBox1.Text + _
        TextBox2.Text
```

Kemudian, jalankan program dan coba isikan variabel 1 dan variabel 2 dengan beberapa kondisi seperti contoh berikut :

- **Jika variabel 1 diisi dengan angka *200* dan variabel 2 diisi dengan angka 300 kemudian dilakukan klik pada tombol *Proses Numerik*, maka hasil akan menunjukkan *500.* Hal ini menunjukkan bahwa dengan menggunakan proses numerik, maka kita yakin bahwa nilai yang diinputkan akan dikonversi terlebih dulu menjadi tipe jenis numerik.**
- **Jika variabel 1 diisi dengan angka *200* dan variabel 2 diisi dengan angka 300 kemudian dilakukan klik pada tombol *Proses Biasa*, maka hasil akan menunjukkan *200300***

Konversi secara eksplisit

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 :

1. **CDate**
   Mengkonversi dari tipe data string ke tipe data tanggal
2. **CInt, Val**
   Mengkonversi dari tipe data string ke tipe data integer
3. **CStr, Str**
   Mengkonversi dari tipe data lain (boolean, date, numerik) ke tipe string
4. **Chr, Chrw**
   Mengkonversi kode angka ASCII ke karakter
5. **Asc, Ascw**
   Kebalikan dari fungsi Chr dan Chrw
6. **Lcase, Ucase**
   Mengkonversi sebuah tipe data string ke huruf besar (Ucase) dan huruf kecil (Lcase)
7. **DateSerial, TimeSerial**

Mengkonversi sebuah serial angka (berupa rangkaian bulan, tanggal dan tahun) ke tipe data date (DateSerial) atau waktu (TimeSerial)

8. **DateValue, TimeValue**

   Hampir sama dengan CDate

9. **Day, Month, Year, Weekday**

   Mengambil nilai tertentu dari tipe data date, misal : fungsi Day untuk mengambil tanggal dan seterusnya.

Soal latihan :

Buat form seperti pada gambar berikut, yang akan mengkonversikan nilai – nilai numerik menjadi sebuah tanggal. Untuk melakukan hal – hal tersebut, dapat digunakan fungsi *DateSerial* yang membutuhkan parameter tahun, bulan dan tanggal.

Gambar 5.4. Form untuk konversi numerik ke tanggal secara eksplisit

Jawab :

Pada form tersebut, terdapat tiga buah komponen *NumericUpDown* yang ditujukan hanya untuk menerima inputan angka. Pada *NumericUpDown* pertama, set property *Minimum* menjadi 1, dan property *Maximum* menjadi 31. Untuk *NumericUpDown* kedua, set property *Minimum* menjadi 1 dan *Maximum* menjadi 12. Sedangkan pada *NumericUpDown* yang terakhir, set property *Minimum* menjadi 1999 dan *Maximum* menjadi 2999.

Setelah itu ketikkan listing berikut pada tombol *Konversi Tanggal*.

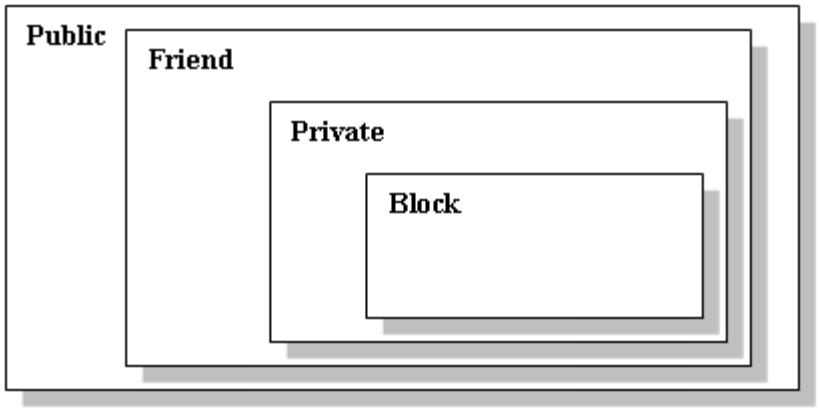
```
MessageBox.Show(DateSerial _
```

```
(NumericUpDown3.Value, NumericUpDown2.Value, _
NumericUpDown1.Value), "Hasil konversi", MessageBoxButtons.OK)
```

Gunakan komponen *NumericUpDown* untuk isian field yang hanya mengizinkan pengisian angka, untuk menghindari kesalahan inputan dari pengguna.

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 :



Gambar 5.5. 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.

2. **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.

3. **Friend**

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

4. **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.

5. **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*

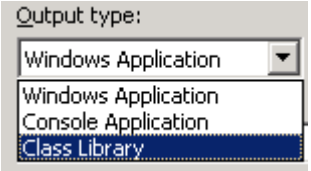
Soal latihan :

Untuk mendemonstrasikan penggunaan variabel dengan jangkauan *Public* di dua project yang berbeda, maka kerjakan soal latihan berikut :

1. **Buat sebuah solution dengan nama *Bab5Variabel* yang nantinya akan menampung dua buah project.**
2. **Buat sebuah project dengan nama *ProjectSatu* yang mempunyai sebuah form bernama *FormSatu* dengan satu variabel public bertipe string. Untuk jangkauan public, variabel harus diletakkan di level teratas sebuah class.**

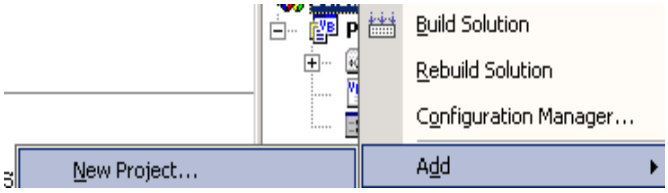
```
Public Class FormSatu
    Inherits System.Windows.Forms.Form
    Public Shared xKata As String = "TEST"
```

3. **Set property project dari *ProjectSatu* menjadi kompilasi untuk tipe *Class Library***



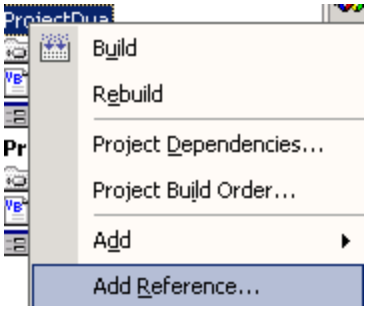
Gambar 5.6. Setting property untuk ProjectSatu

4. **Kemudian buat sebuah project baru lagi di solution yang sama dengan nama *ProjectDua* yang akan mempunyai sebuah form dengan nama *FormDua*. Pembuatan project baru dapat dengan mengklik kanan solution dan mengakses menu *Add New Project***

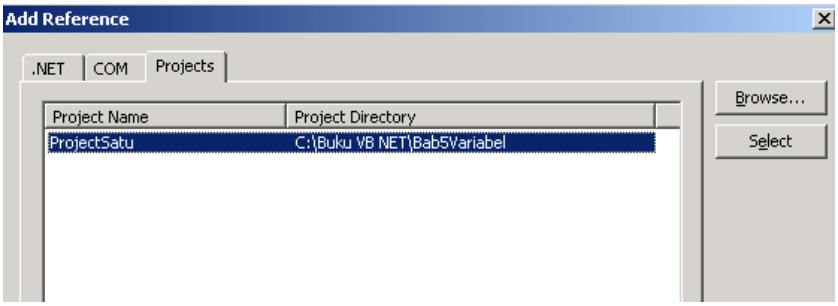


Gambar 5.7. Menu shortcut pembuatan project baru

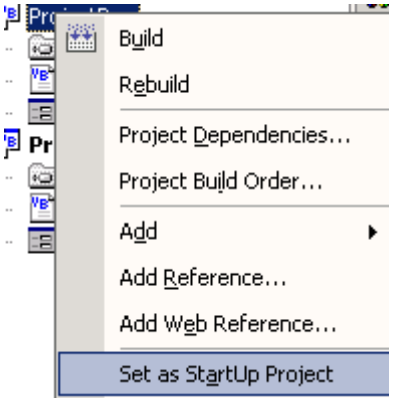
5. **Pada *ProjectDua* tambahkan referensi ke *ProjectSatu* agar dapat memanggil variabel dengan jangkauan public tersebut. Penambahan tersebut dilakukan dengan mengklik kanan pada *ProjectDua* dan mengakses menu Add Reference. Pada *ProjectDua* juga harus diset menjadi project yang akan dijalankan pertama kali dengan menjadikannya sebagai *Startup Project*.**



Gambar 5.8. Menu shortcut untuk penambahan reference



Gambar 5.9. Kotak dialog penambahan reference



Gambar 5.10. Menu shortcut startup project

6. **Di *FormDua,* ketikkan listing berikut pada event *Form_Load* untuk mengakses variabel dengan jangkauan public dari *ProjectSatu***

```
MessageBox.Show(Bab5Variabel.FormSatu.xKata)
```

7. **Jika sudah, eksekusi solution tersebut, maka yang tampil pertama kali adalah form dari *ProjectDua* dengan menampilkan kotak pesan yang berasal dari variabel yang diakses dari *ProjectSatu* dengan menggunakan jangkauan *Public*.**

Perhatikan, penggunaan jangkauan *public* di dua project yang berbeda akan membutuhkan sebuah *reference* dari project yang ada..

If....then ....End if

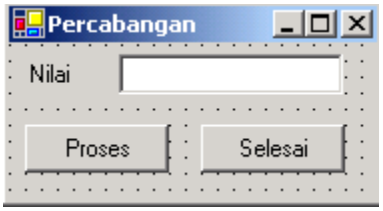
Merupakan percabangan yang umumnya digunakan untuk percabangan dengan kondisi tunggal. Sintaks dari perintah ini adalah sebagai berikut :

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*

Soal latihan :

Buat sebuah form yang akan menyatakan pesan *Lulus* jika nilai yang dimasukkan lebih besar dari 50, dan jika kurang, maka akan menampilkan pesan *Tidak Lulus*. Layout form seperti pada gambar berikut :



Gambar 5.11. Layout soal percabangan

Jawab :

1. Buat sebuah solution baru dengan sebuah form.

2. **Tempatkan satu buah label, satu buah textbox dan dua buah button sehingga layout form menjadi seperti pada gambar.**
3. **Pada button dengan text *Selesai* lakukan klik ganda dan ketikkan listing berikut:**

```
Close()
```

4. **Pada button dengan text *Proses* lakukan klik ganda untuk mengakses event *Click*, lalu ketikkan listing berikut :**

```
If TextBox1.Text > 50 Then
    MessageBox.Show("Lulus", "Hasil")
Else
    MessageBox.Show("Tidak lulus", "Hasil")
End If
```

Soal latihan :

Buat sebuah konversi nilai ke huruf dengan kondisi sebagai berikut :

- **Jika nilai antara 0 - 20 maka dikonversi menjadi E**
- **Jika nilai antara 21 - 40 maka dikonversi menjadi D**
- **Jika nilai antara 41 - 60 maka dikonversi menjadi C**
- **Jika nilai antara 61 - 80 maka dikonversi menjadi B**
- **Jika nilai antara 81 - 100 maka dikonversi menjadi A**

Layout form dibuat sama dengan soal latihan sebelumnya.

Jawab :

1. **Lakukan langkah yang sama dengan soal latihan sebelumnya hingga langkah nomor 3**
2. **Pada button dengan text *Proses* ketikkan listing berikut**

```
Dim x As String
If TextBox1.Text > 0 And TextBox1.Text <= 20 Then x = "E"
If TextBox1.Text > 20 And TextBox1.Text <= 40 Then x = "D"
If TextBox1.Text > 40 And TextBox1.Text <= 60 Then x = "C"
If TextBox1.Text > 60 And TextBox1.Text <= 80 Then x = "B"
If TextBox1.Text > 80 And TextBox1.Text <= 100 Then x = "A"
```

```
MessageBox.Show(x, "Hasil")
```

Anda bisa menggunakan perintah percabangan *if* tanpa penutup *end if*, dengan syarat pernyataan hasil kondisi hanya satu baris.

Checkbox

Checkbox merupakan salah satu contoh komponen yang menggunakan percabangan *if...then...end if*. Komponen ini mempunyai beberapa property yang sering digunakan antara lain :

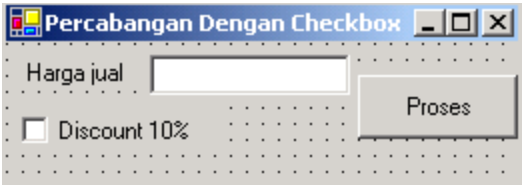
1. **Checked**

   Merupakan property utama yang menangkap nilai dari sebuah checkbox, jika bernilai *True* maka berarti checkbox telah dipilih oleh pengguna, sedangkan jika tidak, maka akan mengembalikan nilai *False*
2. **CheckAlign**

   Mengatur perataan teks dari sebuah checkbox. Apakah teks akan ditempatkan di sebelah kiri atau kanan sebuah checkbox.

Soal latihan :

Buat sebuh form yang akan menghitung harga netto dari sebuah inputan nilai. Harga akan diberi potongan harga sebesar 10% jika checkbox diklik oleh pengguna.



Gambar 5.12. Layout latihan soal

Jawab :

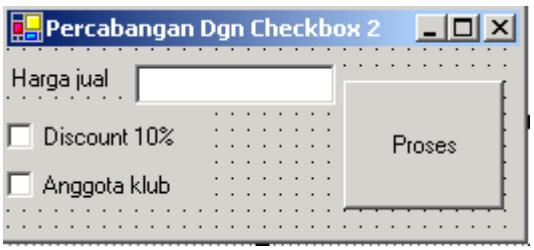
1. **Buat sebuah solution dan project baru**
2. **Di dalam form, tempatkan sebuah label, sebuah textbox dan button serta checkbox. Atur property text masing – masing komponen hingga menjadi seperti pada gambar di soal latihan.**

3. **Pada button, lakukan klik ganda dan ketikkan listing berikut**

```
Dim x As Decimal
x = TextBox1.Text
If CheckBox1.Checked Then x = x * 0.9
MessageBox.Show(x, "Harga Netto")
```

Soal latihan :

Hampir sama dengan soal latihan sebelumnya, tetapi ditambahkan sebuah checkbox lagi yang menyatakan keanggotaan klub, yang jika diklik oleh pengguna, maka harga akan dipotong lagi sebesar 15%



Gambar 5.13. Layout latihan soal

Jawab :

Lakukan langkah yang sama dengan soal latihan sebelumnya, tetapi pada listing, ganti menjadi listing berikut :

```
Dim x, y As Decimal
x = TextBox1.Text
y = 0
If CheckBox1.Checked Then y = x * 0.1
If CheckBox2.Checked Then y = y + (x * 0.15)
MessageBox.Show(x - y, "Harga Jual")
```

Jika terdapat lebih dari satu checkbox dalam sebuah form, dan checkbox – checkbox tersebut saling berkaitan prosesnya, gunakan teknik pemrograman satu *if* untuk tiap checkbox yang ada dengan bantuan variabel tertentu..

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 :

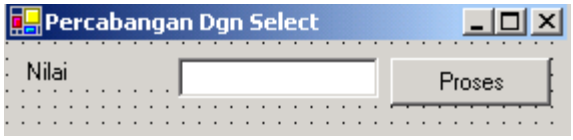
Select [Case] *kondisi*
[Case *daftar* kondisi
[*hasil*]]
[Case Else
[*hasil*]]
End Select

Soal latihan :

Buat sebuah aplikasi untuk menangani proses konversi nilai dengan syarat sebagai berikut :

- **Jika nilai antara 0 - 20 maka dikonversi menjadi E**
- **Jika nilai antara 21 - 40 maka dikonversi menjadi D**
- **Jika nilai antara 41 - 60 maka dikonversi menjadi C**
- **Jika nilai antara 61 - 80 maka dikonversi menjadi B**
- **Jika nilai antara 81 - 100 maka dikonversi menjadi A**

Buat layout form seperti pada gambar berikut :



Gambar 5.14 Layout soal latihan

Jawab :

Pada button *Proses* ketikkan listing berikut :

```
Dim x As String
Select Case Val(TextBox1.Text)
    Case 0 To 20
        x = "E"
```

```
    Case 21 To 40
        x = "D"
    Case 41 To 60
        x = "C"
    Case 61 To 80
        x = "B"
    Case 81 To 100
        x = "A"
    Case Else
        x = "Tidak terdefinisi !"
End Select
MessageBox.Show(x, "Hasil konversi")
```

þ Keterangan

Pada percabangan dengan menggunakan select case ... end select, sebuah jangkauan nilai yang urut (baik angka ataupun karakter) dapat direpresentasikan dengan kata *to*. Sehingga sebuah jangkauan nilai dari 0 hingga 20 dapat diwakili dengan kata *0 to 20.*

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*]

Soal latihan :

Buat form dengan layout seperti pada gambar, dengan kondisi sebagai berikut :

- **Tombol *Genap* akan menampilkan bilangan genap hingga batas yang tertera di *NumericUpDown*.**
- **Tombol *Ganjil* akan menampilkan bilangan ganjil hingga batas yang tertera di *NumericUpDown*.**
- **Tombol *Mundur* akan menampilkan bilangan hingga batas yang tertera di *NumericUpDown* dengan urutan terbalik (dari besar ke kecil atau descending)**
- **Tombol *Fibonacci* akan menampilkan deret bilangan Fibonacci hingga batas yang tertera di *NumericUpDown*. Deret Fibonacci merupakan deret bilangan yang menjumlahkan dua angka sebelumnya pada deret tersebut, dengan angka inisial 0 dan 1 pada awal deret bilangan. Misal : 0,1,1,2,3,5,8,13 dst.**

Gambar 5.15. Layout latihan perulangan

Jawab :

Sebelum mengetikkan listing program, terlebih dulu set property *Minimum* untuk *NumericUpDown* menjadi 2 dan property *Maximum* menjadi 100. Sedangkan untuk textbox sebagai penampung hasil, set property *Multiline* menjadi *True* agar bisa menampung hasil lebih dari satu baris.

Untuk tombol *Genap* ketikkan listing berikut :

```
TextBox1.Clear()
For x As Integer = 1 To NumericUpDown1.Value
    If x Mod 2 = 0 Then
        TextBox1.AppendText (x & vbCrLf)
    End If
Next
```

Untuk tombol *Ganjil* ketikkan listing berikut :

```
TextBox1.Clear()
For x As Integer = 1 To NumericUpDown1.Value
    If x Mod 2 > 0 Then
        TextBox1.AppendText (x & vbCrLf)
    End If
Next
```

þ Keterangan

Kata kunci *Mod* merupakan fungsi yang akan menghasilkan sisa hasil bagi. Misal : 2 Mod 1 akan menghasilkan angka 1. Sehingga jika hasil dari sebuah bilangan Mod 2 lebih besar dari 1 akan menghasilkan sebuah bilangan ganjil, dan sebaliknya akan menghasilkan bilangan genap.

Untuk tombol *Mundur*, listingnya adalah :

```
TextBox1.Clear()
For x As Integer = NumericUpDown1.Value To 1 Step -1
    TextBox1.AppendText(x & vbCrLf)
Next
```

þ Keterangan

Untuk menghasilkan perulangan dari bilangan besar ke bilangan kecil (descending) maka ditambahkan kata *Step* dengan perhitungan mundur atau bilangan negatif.

Untuk menampilkan *Fibonacci*, bisa diketikkan listing berikut :

```
Dim w, y, z As Integer
y = 0
z = 1
TextBox1.Clear()
TextBox1.AppendText (y & vbCrLf & z & vbCrLf)
For x As Integer = 1 To NumericUpDown1.Value
    w = y + z
    y = z
    z = w
    If w > NumericUpDown1.Value Then Exit For
    TextBox1.AppendText(w & vbCrLf)
Next
```

þ Keterangan

Perintah *Exit For* akan mengakibatkan perulangan terhenti meski belum mencapai batas yang sudah ditentukan.

Sintaks dari perintah ini adalah :

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

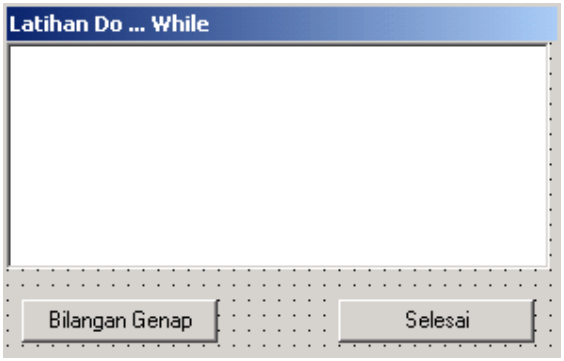
Atau

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).

Soal latihan :

Buat sebuah form seperti pada gambar berikut untuk menampilkan deret bilangan genap dari 1 hingga 10 secara menurun pada sebuah textbox.



Gambar 5.16 Layout form soal latihan

Jawab :

1. **Pada textbox, set property *MultiLine* menjadi *True***
2. **Kemudian pada button *Bilangan Genap* ketikkan listing berikut :**

```
Dim x, y As Integer
x = 0 : y = 10
Do While x <= y
    If x Mod 2 = 0 Then TextBox1.Text &= " " & x & vbCrLf
    x += 1
Loop
```

þ Keterangan

Dengan menggunakan perintah *Do.. While*, maka logika pemrograman sedikit berbeda dengan implementasi pada perintah *For .. Next*, yaitu dengan adanya variabel x sebagai penghitung atau counter dan variabel y sebagai pembatas dari perulangan tersebut. Dan baris *x+=1* merupakan baris perintah yang dibutuhkan agar perulangan berjalan secara terkendali, tidak berulang terus tanpa batas yang jelas. Dengan kata lain, perintah *x+=1* akan memerintahkan agar variabel x terus ditambahkan dengan angka satu hingga perulangan berhenti.

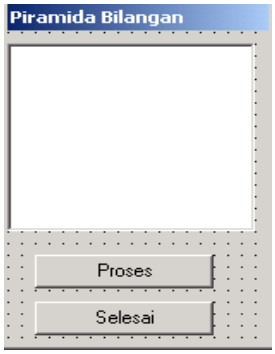
Soal latihan :

Buat sebuah piramida bilangan dengan format sebagai berikut :

```
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

Jawab :

1. **Buat sebuah form dengan layout seperti pada gambar berikut (dua buah button dan satu buah textbox). Pada komponen textbox, set property *MultiLine* menjadi *true* dan property *font* ganti dengan jenis *courier new*.**



Gambar 5.17 Layout piramida bilangan

2. **Pada tombol *Proses* ketikkan listing berikut :**

```
Dim x, y, z As Integer
x = 1 : y = 5
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
```

þ Keterangan

Fungsi *space* merupakan sebuah fungsi yang menempatkan spasi kosong sebanyak parameter angka yang terdapat dalam fungsi tersebut.
Pada contoh soal latihan, didemonstrasikan penggunaan sebuah *nested loop* atau perulangan bersarang, yaitu perulangan yang terdapat dalam perulangan yang lain.
Yang perlu diperhatikan dalam sebuah *nested loop* adalah bahwa masing – masing perulangan memiliki penghitung tersendiri dan tidak berbenturan satu sama lain, sehingga masing – masing perulangan tidak nejadi sebuah *continuos looping* atau sebuah perulangan yang tidak pernah berhenti.
Pemakaian jenis font *courier new* pada textbox akan menyebabkan penempatan huruf atau angka menjadi lebih rata, dibanding dengan jenis font yang lain.

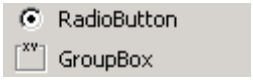
For Each ……. Next

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
```

GroupBox dan Radiobutton

Penggunaan komponen radiobutton seringkali ditempatkan dalam sebuah groupbox. Hal ini untuk mempermudah pengelompokan sebuah grup dari radiobutton.



Gambar 5.18 Komponen groupbox dan radiobutton dalam toolbox

Sebuah groupbox merupakan komponen yang secara logika akan mengelompokkan beberapa komponen lain dalam sebuah *controls collection*. Sedangkan radiobutton adalah komponen yang secara umum digunakan untuk menghasilkan suatu pilihan tunggal dalam sebuah pilihan majemuk (multiple choice).

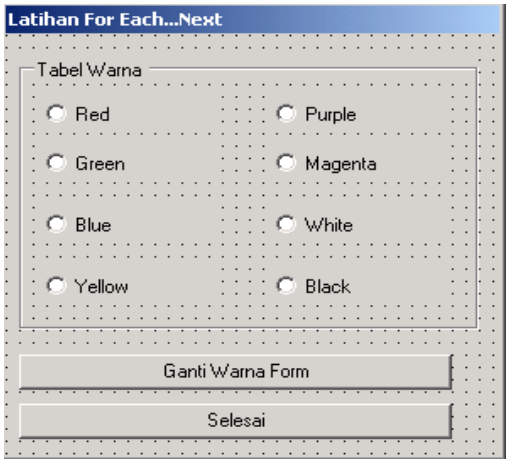
Dalam komponen radiobutton, terdapat beberapa property yang paling sering dipakai yaitu :

1. **Checked**

   Menentukan nilai dari suatu radiobutton jika dklik oleh pengguna. Mempunyai dua nilai : true dan false
2. **Text**

Soal latihan :

Buat sebuah form yang memiliki fungsi untuk mengganti warna latar belakang sebuah form, dengan memanfaatkan komponen groupbox dan radiobutton serta perintah *for each….next*



Gambar 5.19 Layout form ganti warna

Jawab :

1. **Buat form seperti pada gambar**
2. **Pastikan bahwa komponen radiobutton diletakkan di dalam komponen groupbox.**

Agar radiobutton terletak di dalam groupbox, maka saat akan menempatkan radiobutton, pastikan bahwa groupbox dalam keadaan terseleksi dengan melakukan klik kiri satu kali pada groupbox sebelum menempatkan radibutton..

3. **Untuk property *text* dari radiobutton harus sama persis seperti pada gambar untuk mengaplikasikan penggunaan perintah *color***
4. **Pada button *Ganti Warna Form* ketikkan listing berikut :**

```
For Each i As RadioButton In GroupBox1.Controls
    If i.Checked = True Then
        Me.BackColor = Color.FromName(i.Text)
    End If
Next
```

þ Keterangan

Pada contoh perintah di atas perulangan akan dilakukan sebanyak jumlah radiobutton yang terdapat di dalam groupbox. Dengan mendefinisikan variabel i sebagai sebuah variabel dengan tipe radiobutton, maka dengan leluasa kita dapat mengambil property text dari radiobutton yang sedang diklik oleh pengguna untuk menjadi umpan balik bagi warna latar belakang form.
Variabel i selain sebagai tipe radiobutton juga bisa didefinisikan sebagai sebuah variabel dengan tipe *object* jika terdapat beragam tipe komponen yang akan dimasukkan ke dalam perulangan.

Soal latihan :

Sama dengan soal latihan sebelumnya, tetapi property text di masing – masing radiobutton diganti dengan bahasa Indonesia.

Gambar 5.20 Layout form ganti warna dengan bahasa Indonesia

Jawab :

1. **Lakukan langkah yang sama seperti pada soal latihan sebelumnya**
2. **Pada masing – masing radiobutton, ganti nilai yang terdapat dalam property *tag* dengan nama warna dalam bahasa Inggris.**

Gambar 5.21 Contoh penggantian nilai property tag

3. Kemudian pada button *Ganti warna form* ketikkan listing berikut :

```
For Each i As RadioButton In GroupBox1.Controls
    If i.Checked = True Then
        Me.BackColor = Color.FromName(i.Tag)
    End If
Next
```

þ Keterangan

Property *tag* merupakan property yang tidak memiliki fungsi khusus di dalam Visual Basic .NET ataupun dalam bahasa pemrograman visual yang lain. Property ini akan memiliki nilai yang bertipe string. Dan seringkali digunakan sebagai *tanda khusus* dalam suatu komponen untuk memudahkan pemrograman, seperti halnya pada contoh soal latihan.

Soal latihan :

Buat form seperti pada gambar, dan jika tombol *Hitung Pesanan* ditekan, maka akan dilakukan perhitungan harga makanan dan minuman. Manfaatkan property tag dan perintah *for each* , agar listing yang dihasilkan dapat seminimal mungkin.

Gambar 5.22 Layout soal form dengan dua groupbox

Jawab :

1. **Buat layout form seperti pada gambar. Perlu diperhatikan agar peletakan groupbox pada form dilakukan terlebih dulu, sebelum peletakan radiobutton yang harus diletakkan di dalam groupbox tersebut.**
2. **Pada property masing – masing radiobutton, set property *Text* dan *Tag* seperti pada gambar berikut :**

Gambar 5.23 Pemberian nilai pada property tag

3. **Pada button *Hitung Pesanan* ketikkan listing berikut :**

```
Dim x, y As Integer
For Each i As RadioButton In GroupBox1.Controls
    If i.Checked = True Then
        x = i.Tag
    End If
Next
For Each i As RadioButton In GroupBox2.Controls
    If i.Checked = True Then
        y = i.Tag
    End If
Next
MessageBox.Show("Total : Rp. " & _
    Format(x + y, "###,###"), "Hasil")
```

Gunakan property tag pada komponen yang akan dicek sebagai sebuah *collection,* untuk mempermudah teknik pemrograman.

Listbox & Combobox

Listbox dan combobox merupakan dua komponen yang mempunyai fungsi dan sifat yang mirip. Kedua komponen tersebut sama – sama menawarkan pilihan kepada user, perbedaan yang menyolok adalah listbox dapat dipilih lebih dari satu pilihan oleh pengguna, sedangkan di dalam combobox pengguna dipaksa untuk memilih satu item saja.

Beberapa property yang sering dipakai di dalam listbox dan combobox adalah :

1. **Items**

Merupakan property utama dalam listbox dan combobox yang menampung semua item yang akan ditampilkan.

2. **Text**

   Property yang menunjukkan item yang sedang aktif atau sedang dipilih.

3. **SelectedIndex**

   Property yang menunjukkan posisi item yang sedang aktif atau sedang dipilih dan dimulai dari angka 0.

4. **SelectedText**

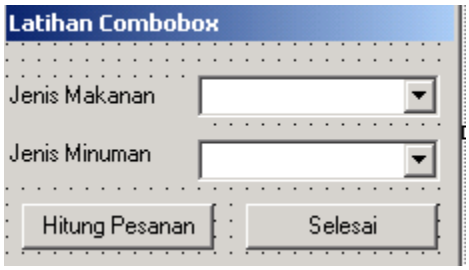
   Fungsinya hampir sama dengan property text

5. **MultiSelect**

   Khusus untuk listbox, menyatakan status listbox yang dapat dipilih itemnya lebih dari satu dalam satu kali pilihan. Memilih lebih dari satu dapat dilakukan dengan klik kiri yang dikombinasikan dengan penekan tombol shift atau ctrl.

Soal latihan :

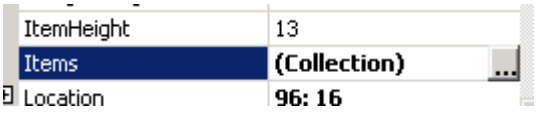
Buatlah form dengan kasus yang sama pada soal latihan terakhir, tetapi dengan menggunakan komponen combobox.



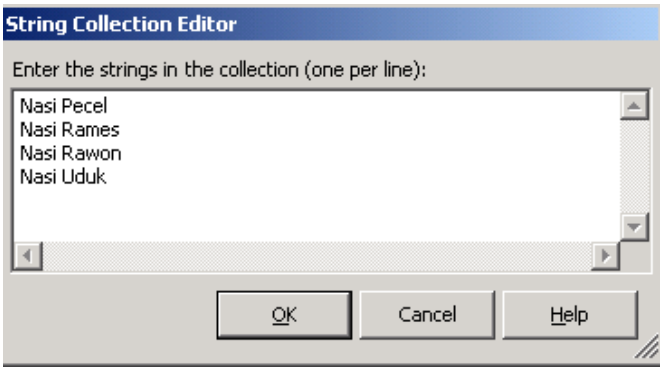
Gambar 5.23 Layout soal combobox

Jawab :

1. **Buat layout form seperti pada gambar**
2. **Pada combobox pertama untuk jenis makanan, set property items dengan mengisikan masing – masing item langsung saat design time melalui string collection editor. Kotak dialog string colletion editor dapat diakses dengan cara menekan tombol kecil di sebelah kanan property items.**

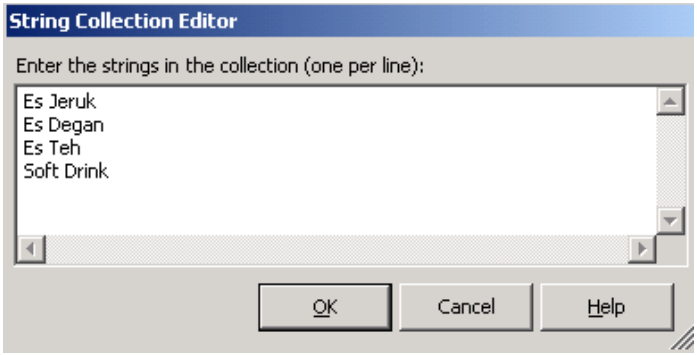


Gambar 5.24 Property items pada combobox



Gambar 5.25 Setting nilai property items pada combobox pertama

3. **Cara yang sama juga dilakukan untuk combobox yang kedua yaitu berisi tentang jenis minuman.**



Gambar 5.26 Setting nilai property items pada combobox kedua

1. **Agar pada saat program dijalankan kedua combobox tidak menampilkan isi kosong, maka pada *form_load* ketikkan listing berikut :**

```
ComboBox1.SelectedIndex = 0
ComboBox2.SelectedIndex = 0
```

2. **Pada tombol *Hitung Pesanan* ketikkan listing berikut :**

```
Dim x, y As Integer
Select Case ComboBox1.Text
    Case "Nasi Pecel"
        x = 2500
    Case "Nasi Rames"
        x = 3500
```

```
    Case "Nasi Rawon"
        x = 3000
    Case Else
        x = 2750
End Select
Select Case ComboBox2.Text
    Case "Es Teh"
        y = 1000
    Case "Es Degan", "Es Jeruk"
        y = 1500
    Case Else
        y = 2000
End Select
MessageBox.Show("Total : Rp. " & _
    Format(x + y, "###,###"), "Hasil")
```

þ Keterangan

Property *Text* merupakan property yang menunjukkan nilai yang sedang dipilih oleh pengguna di dalam sebuah combobox.
Sedangkan property *selectedIndex* menunjukkan indeks item yang sedang ditampilkan oleh combobox tersebut. Jika property *selectedIndex* diberi nilai 0, maka combobox akan menampilkan item pertama dari koleksi item yang ada.

Soal latihan :

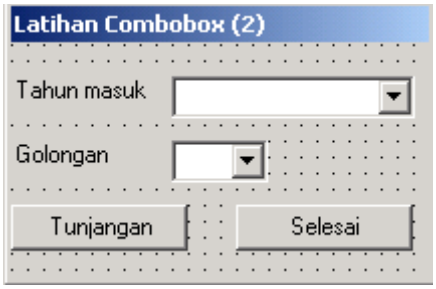
Buat sebuah perhitungan untuk total tunjangan yang merupakan hasil penjumlahan dari tunjangan masa kerja dan tunjangan golongan dengan kondisi sebagai berikut :

- **Jika masuk pada tahun 2000, maka mendapat tunjangan masa kerja sebesar Rp. 300.000**
- **Jika masuk pada tahun 2001, maka mendapat tunjangan masa kerja sebesar Rp. 250.000**
- **Jika masuk pada tahun 2002, maka mendapat tunjangan masa kerja sebesar Rp. 200.000**
- **Jika masuk pada tahun 2003, maka mendapat tunjangan masa kerja sebesar Rp. 150.000**

Sedangkan untuk masing – masing golongan mendapat tunjangan sebesar :

- **Golongan A sebesar Rp. 200.000**
- **Golongan B sebesar Rp. 150.000**
- **Golongan C sebesar Rp. 100.000**

Isikan nilai yang ada dalam combobox melalui *run time*, tidak melalui kotak property items collection.



Gambar 5.27 Layout soal combobox kedua

Jawab :

1. **Buat layout form seperti pada gambar**
2. **Untuk mengisikan nilai item pada masing – masing combobox, ketikkan listing berikut pada *form_load***

```
For i As Integer = 2000 To 2004
    ComboBox1.Items.Add(i)
Next
ComboBox1.SelectedIndex = 0
For i As Integer = 65 To 67
    ComboBox2.Items.Add(Chr(i))
Next
ComboBox2.SelectedIndex = 0
```

3. **Sedangkan untuk menyelesaikan perhitungan tunjangan, pada button *Tunjangan* ketikkan listing berikut :**

```
Dim x, y As Integer
Select Case ComboBox1.Text
    Case "2000"
        x = 300000
    Case "2001"
        x = 250000
    Case "2002"
        x = 200000
    Case "2003"
```

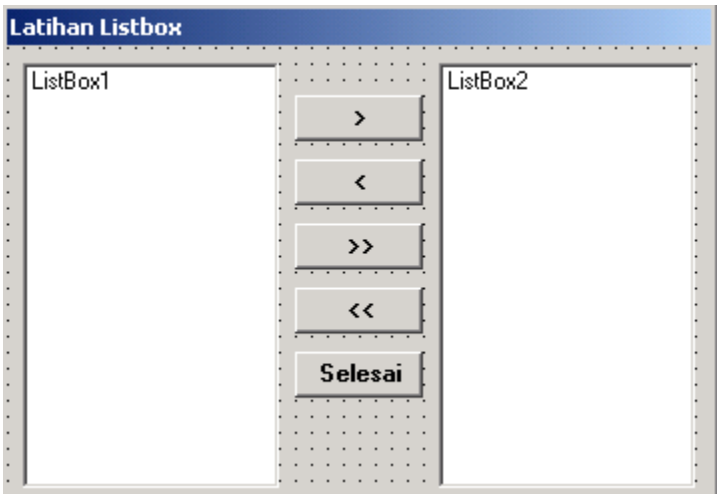
```
            x = 150000
        Case "2004"
            x = 0
    End Select
    Select Case ComboBox2.Text
        Case "A"
            y = 200000
        Case "B"
            y = 150000
        Case "C"
            y = 100000
    End Select
    MsgBox(ComboBox2.SelectedText)
    MessageBox.Show("Tunjangan : " & _
        Format(x + y, "#,###,###"), "Hasil")
```

þ Keterangan

Pengisian item pada combobox pada saat *run time* dilakukan dengan mengetikkan *ComboBox1.Items.Add(i)*, dengan i sebagai parameter yang berisikan tipe data string. Sedangkan property *SelectedIndex* digunakan untuk menampilkan item dengan item tertentu pada saat form pertama kali muncul. Pada contoh soal, property tersebut diberi nilai nol yang merupakan nilai awal dari suatu collection item di dalam combobox.

Soal latihan :

Buatlah sebuah form seperti pada gambar berikut, yang mendemonstrasikan perpindahan nilai item dari listbox ke sebelah kiri ke sebelah kanan dan sebaliknya.



Gambar 5.28 Layout soal listbox

Jawab :

1. **Buat form seperti pada gambar**
2. **Pertama, isikan terlebih dulu nilai awal di listbox sebelah kiri dengan angka dari 0 hingga 9. Tempatkan listing untuk pengisian listbox di dalam *Form_Load*.**

```
For i As Integer = 0 To 9
   ListBox1.Items.Add("Item ke : " & i)
Next
```

3. **Pada button untuk memindah ke kiri satu item (>), tempatkan listing berikut :**

```
If ListBox1.SelectedItem <> "" Then
    ListBox2.Items.Add _
        (ListBox1.SelectedItem)
    ListBox1.Items.Remove _
        (ListBox1.SelectedItem)
End If
```

þ Keterangan

Property *SelectedItem* menunjukkan item yang sedang dipilih oleh pengguna di dalam listbox. Jika property tersebut tidak memiliki nilai, maka berarti tidak ada item yang dipilih oleh pengguna, sehingga tidak ada yang perlu dilakukan dalam prosedur.
Logika dari listing program tersebut adalah memindah terlebih dulu item yang sedang dipilih oleh pengguna ke listbox sebelah kanan, kemudian item yang sedang dipilih tersebut dihapus dari listbox yang bersangkutan. Logika yang sama juga diterapkan pada button untuk pindah ke kanan satu item.
Pastikan bahwa kedua listbox untuk property *Sorted* bernilai *true* agar saat terjadi proses pemindahan, urutan item tetap terjaga.

4. **Pada button untuk memindah ke kanan satu item (<), tempatkan listing berikut :**

```
If ListBox2.SelectedItem <> "" Then
    ListBox1.Items.Add _
        (ListBox2.SelectedItem)
    ListBox2.Items.Remove _
        (ListBox2.SelectedItem)
End If
```

5. **Pada button untuk memindah ke kiri semua item (>>), tempatkan listing berikut :**

```
ListBox1.Items.AddRange(ListBox2.Items)
ListBox2.Items.Clear()
```

þ Keterangan

Method *AddRange* memindahkan sebuah grup (dalam soal latihan adalah listbox sebelah kiri) ke listbox yang lain.

6. Pada button untuk memindah ke kanan semua item (<<), tempatkan listing berikut :

```
ListBox1.Items.AddRange(ListBox2.Items)
ListBox2.Items.Clear()
```

Ingatlah bahwa pemakaian *AddRange* akan mengisikan atau memindahkan seluruh isi item dalam sebuah listbox.

Soal latihan :

Buat sebuah form sederhana untuk menampilkan tabel cicilan dari sebuah pinjaman dengan menggunakan perhitungan bunga yang menurun. Layout awal dari form tersebut adalah sebagai berikut :

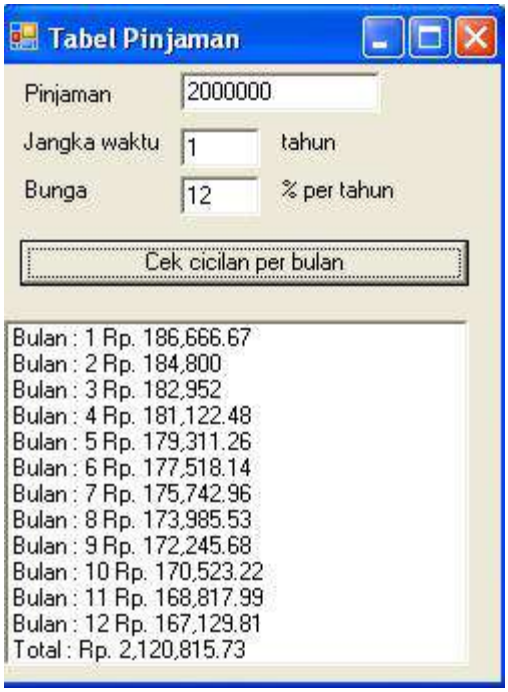
Gambar 5.29. Layout Awal Form Tabel Pinjaman

Jawab :

1. **Buat form seperti pada gambar**
2. **Pada button *Cek Cicilan* ketikkan listing berikut ini :**

```
Dim j, k, l, m As Decimal
ListBox1.Items.Clear()
j = TextBox1.Text
For i As Integer = 1 To (CInt(TextBox2.Text) * 12)
    k = (TextBox1.Text / 12) + _
        ((j * (TextBox3.Text / 100)) / 12)
    j -= k
    l += k
    ListBox1.Items.Add("Bulan : " & i & " Rp. " & _
                       Format(k, "###,###.##"))
    m += k
Next
ListBox1.Items.Add("Total : Rp. " & _
                   Format(m, "###,###,###.##"))
```

3. **Jika form telah dijalankan maka akan tampak tampilan seperti pada gambar berikut :**



Gambar 5.30. Tampilan akhir

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**

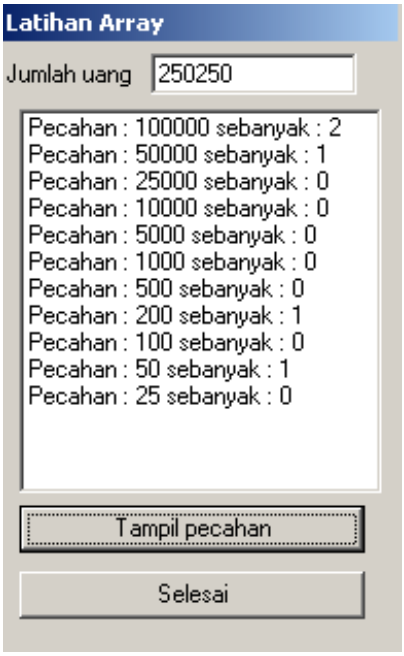
   Merupakan array yang hanya memiliki satu urutan indeks
2. **Aray dua dimensi**

   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*.

Soal latihan :

Buatlah form seperti pada gambar untuk menentukan pecahan uang yang harus dibayarkan berdasarkan pecahan mata uang rupiah yang ada saat ini.



Gambar 5.31 Layout soal array

Jawab :

1. **Buat form seperti pada gambar**
2. **Di dalam button *Tampil pecahan*, ketikkan listing berikut :**

```
Dim xpecahan() As Integer = {100000, 50000, 25000, 10000, _
     5000, 1000, 500, 200, 100, 50, 25}
Dim xhasil, xtemp, i As Integer
xhasil = TextBox1.Text
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
```

Soal latihan :

Dibuat sebuah form untuk menentukan sifat seseorang berdasarkan tanggal lahir. Penentuan sifat dengan menggunakan teknik numerologi yang menghitung angka-angka dalam tanggal lahir untuk menjadi satu digit dan kemudian dipetakan ke dalam sifat-sifat yang ada dalam sebuah array.

Form yang dibuat memiliki layout seperti pada gambar berikut :

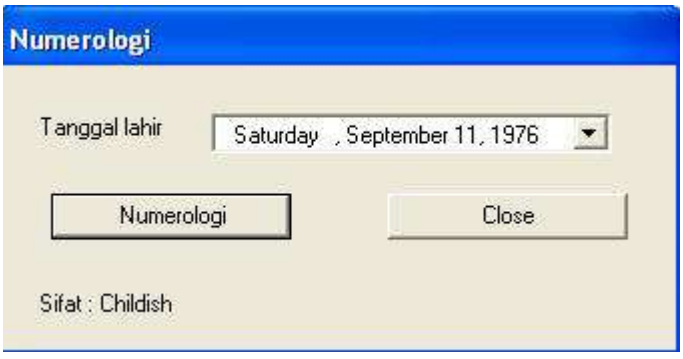
Gambar 5.32 Layout awal form

Jawab :

1. **Buat form seperti pada gambar**
2. **Di dalam button *Numerologi* ketikkan listing berikut ini :**

```
Dim xtemp As Integer = DateTimePicker1.Value.Year + _
    DateTimePicker1.Value.Month + _
    DateTimePicker1.Value.Day
Dim ytemp As String = CStr(xtemp)
Do While ytemp.Length > 1
    Dim ztemp As Integer = 0
    For i As Integer = 1 To ytemp.Length
        ztemp += CInt(Mid(ytemp, i, 1))
    Next
    ytemp = ztemp
Loop
Dim xsifat() As String = Split("Humanis," & _
        "Emosional,Komunikatif,Ramah,Introvert," & _
        "Ekstrovert,Childish,Leader,Humoris", ",")
Label2.Text = "Sifat : " & xsifat(ytemp - 1)
```

3. **Pada saat program dijalankan akan tampil seperti contoh berikut :**



Gambar 5.33 Hasil akhir

Prosedur dan Fungsi

Tujuan :

- **Mampu mengaplikasikan secara tepat dan cermat penggunaan prosedur dan fungsi**
- **Memahami penggunaan handles dalam Visual Basic .NET**

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
```

- **Keterangan**

  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.

Soal latihan :

Buat form seperti pada gambar, kemudian buat prosedur untuk melakukan pengecekan terhadap masing – masing textbox agar diisi dengan nilai numerik.

Gambar 6.1 Layout soal prosedur

Jawab :

1. **Buat form seperti pada gambar**
2. **Buat sebuah prosedur untuk melakukan pengecekan bahwa textbox diisi dengan nilai numerik**

```
Private Sub cekAngka (ByVal xteks As TextBox)
    If Not IsNumeric(xteks.Text) Then
        MessageBox.Show("Harus "  & "diisi dengan angka !")
        xteks.Focus()
    End If
End Sub
```

3. **Panggil prosedur tersebut dalam masing – masing method textbox yang tersedia (textbox pertama hingga ketiga)**

```
Private Sub TextBox1_LostFocus (ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles TextBox1.LostFocus
    cekAngka(sender)
End Sub
```

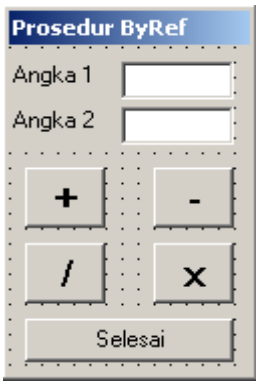
þ Keterangan

Sender merupakan parameter default yang menyatakan obyek yang sedang aktif dalam sebuah method, dan berguna jika sebuah method diarahkan ke banyak control yang berbeda sekaligus. Parameter sender akan dijelaskan lebih lanjut di sub bab berikutnya yaitu tentang *handles*

4. **Jika program dieksekusi, cobalah untuk mengisikan data bukan angka di masing – masing textbox secara bergantian untuk mengamati perilaku dari prosedur yang sudah dibuat.**

Soal latihan :

Buat sebuah kalkulator sederhana dengan layout form seperti pada gambar berikut :



Gambar 6.2 Layout soal prosedur dengan parameter ByRef

Jawab :

1. **Buat form seperti pada gambar**
2. **Buat sebuah prosedur untuk melakukan perhitungan matematika :**

```
Private Sub matematika (ByVal bil1 As Integer, _
     ByVal bil2 As Integer, ByRef hasil As Integer, _
     ByVal operasi As String)
    Select Case operasi
        Case "+"
            hasil = bil1 + bil2
        Case "-"
            hasil = bil1 - bil2
        Case "/"
            hasil = bil1 / bil2
        Case "x"
            hasil = bil1 * bil2
    End Select
End Sub
```

3. **Di dalam button untuk penjumlahan ketikkan listing :**

```
Dim xhasil As Integer = 0
matematika(TextBox1.Text, TextBox2.Text, xhasil, "+")
MessageBox.Show(xhasil)
```

4. **Sesuaikan parameter *operasi* untuk button yang lain**
5. **Eksekusi program dan cek hasilnya**
6. **Edit parameter *hasil* di dalam prosedur *matematika* menjadi parameter *ByVal*, kemudian eksekusi program sekali lagi. Lihat dan bandingkan hasilnya.**

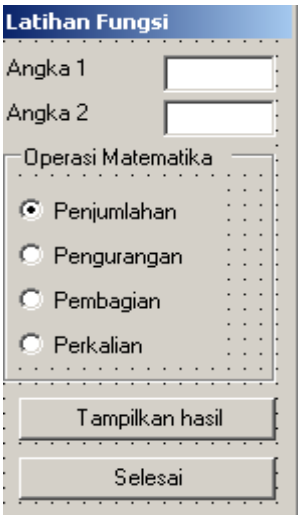
Jika parameter hasil diganti menjadi *ByVal*, maka semua hasil operasi matematika akan tetap bernilai nol, karena variabel *xhasil* di tiap button tidak akan berubah nilainya dari yang asli.

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.

Soal latihan :

Buatlah sebuah kalkulator sederhana dengan memanfaatkan fungsi dalam Visual Basic .NET dan radiobutton.



Gambar 6.3 Layout soal fungsi

Jawab :

1. **Buat form seperti pada gambar**
2. **Buatlah sebuah fungsi untuk menangani operasi matematika dengan parameter dua bilangan hasil inputan serta jenis operasi matematika sesuai dengan radiobutton yang dipilih**

```
Private Function matematika (ByVal bil1 As Integer, _
  ByVal bil2 As Integer, ByVal operasi As RadioButton) _
      As Integer
  Select Case operasi.Text
      Case "Penjumlahan"
          matematika = bil1 + bil2
      Case "Penguranga"
          matematika = bil1 - bil2
      Case "Pembagian"
          matematika = bil1 / bil2
      Case "Perkalian"
          matematika = bil1 * bil2
  End Select
End Function
```

þ Keterangan

Pengumpanan nilai balik bisa dilakukan dengan mengarahkan nama fungsi itu sendiri sebagai hasil suatu perintah, atau juga dengan menggunakan kata kunci *return*, misal : pada contoh fungsi dituliskan *matematika = bil1 + bil2* bisa juga ditulis dengan *return bil1 + bil2*

3. Di dalam button *Tampilkan hasil* ketikkan listing berikut :

```
Dim xradio As RadioButton
  Dim hasil As Integer
  For Each i As RadioButton In GroupBox1.Controls
      If i.Checked = True Then xradio = i
  Next
  hasil = matematika _
      (TextBox1.Text,TextBox2.Text, xradio)
  MessageBox.Show(hasil)
```

Soal latihan :

Buat program untuk melakukan enkripsi teks sederhana dengan layout form seperti pada gambar berikut :

Gambar 6.4 Layout soal fungsi enkripsi

Jawab :

1. Buat form seperti pada gambar

2. **Buat fungsi untuk mengenkripsi teks per karakter, dengan cara menambahkan satu nilai ke kode ASCII dari karakter yang ada di teks normal**

```
Private Function enkrip (ByVal teks As String) As String
    enkrip = Chr(Asc(teks) + 1)
End Function
```

3. **Aplikasikan fungsi ke dalam button *Enkrip Teks***

```
TextBox2.Text = ""
For i As Integer = 1 To Len(TextBox1.Text)
    TextBox2.Text += enkrip(Mid(TextBox1.Text, i, 1))
Next
```

þ Keterangan

Fungsi *Chr* merupakan fungsi untuk mengkonversi suatu kode ASCII menjadi karakter. Sedangkan fungsi *Asc* merupakan kebalikan dari fungsi *Chr* yaitu mengkonversi karakter menjadi kode ASCII.
Sedangkan fungsi *Mid* merupakan fungsi untuk mengambil sejumlah karakter tertentu dari suatu kalimat atau kata. Sintaks dari perintah *Mid* adalah *Mid(kalimat atau kata, posisi karakter yang akan diambil, jumlah karakter yang akan diambil)*.
Dari contoh program tersebut, dapat dibaca bahwa perulangan yang ada akan membaca semua karakter yang ada dalam textbox satu per satu, untuk kemudian diumpankan ke fungsi *enkrip*.

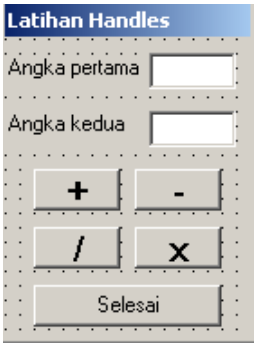
Proses enkripsi yang dicontohkan merupakan proses enkripsi yang sangat sederhana. Visual Basic .NET sendiri memiliki fungsi *built-in* untuk melakukan enkripsi level tinggi seperti RSA dan DES.

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.

Soal latihan :

Buatlah kalkulator sederhana seperti latihan dalam sub bab prosedur, tetapi kali ini memanfaatkan teknik *handles* dalam Visual Basic .NET



Gambar 6.5 Layout soal handles

Jawab :

1. **Buat form seperti pada gambar**
2. **Di dalam button untuk penjumlahan ketikkan listing sebagai berikut :**

```
Private Sub Button1_Click (ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click, _
    Button2.Click, Button3.Click, Button4.Click
    Dim hasil As Integer = 0
    Dim bil1 As Integer = TextBox1.Text
    Dim bil2 As Integer = TextBox2.Text
    Select Case sender.text
        Case "+"
```

```
            hasil = bil1 + bil2
        Case "-"
            hasil = bil1 - bil2
        Case "/"
            hasil = bil1 / bil2
        Case "x"
            hasil = bil1 * bil2
    End Select
    MessageBox.Show(hasil)
End Sub
```

Perhatikan bahwa prosedur untuk button yang menangani penjumlahan juga menangani seluruh button untuk operasi matematika sekaligus. Untuk membedakan antara satu button dengan button yang lain digunakan property *text* sebagai ciri tiap button.

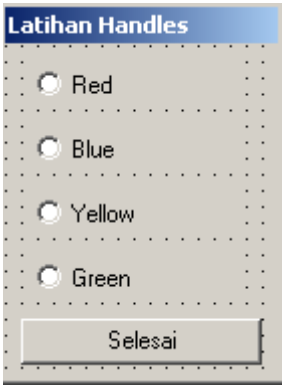
þ Keterangan

Parameter sender dalam sebuah method merupakan representasi dari obyek yang sedang aktif. Jika sebuah method hanya menangani satu control, maka sender adalah control itu sendiri. Tetapi pada kasus soal latihan tersebut, sender akan berlaku sebagai wakil dari control yang sedang aktif, yaitu dari button1 hingga button4, tergantung dari button mana yang sedang diklik oleh pengguna.

Soal latihan :

Buatlah sebuah form yang terdiri dari empat radiobutton untuk mendemonstrasikan penggunaan handles, yaitu tiap radiobutton akan mengganti warna latar belakang form sesuai dengan property text tiap radiobutton.

Penggunaan radiobutton dalam soal latihan ini tidak menggunakan groupbox, agar efek dari perubahan warna form terlihat lebih jelas.



Gambar 6.6 Layout soal handles kedua

Jawab :

1. **Buat form seperti pada gambar**
2. **Pada radiobutton pertama ketikkan listing berikut :**

```
Private Sub RadioButton1_CheckedChanged _
    (ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
  Handles RadioButton1.CheckedChanged, _
    RadioButton2.CheckedChanged, _
    RadioButton3.CheckedChanged, _
    RadioButton4.CheckedChanged
    Me.BackColor = _
          Color.FromName(sender.text)
End Sub
```

Saat anda mengetikkan sender dan diikuti dengan tanda titik, maka fasilitas autocomplete dari Visual Basic .NET tidak akan keluar. Hal ini disebabkan sender sendiri bertipe object, sehingga property maupun method yang ada disesuaikan dengan control yang akan diaktifkan.

Menu dan Form Majemuk

Tujuan :

- **Mampu mengaplikasikan komponen menu dalam Visual Basic .NET**
- **Mampu mengembangkan aplikasi menggunakan form dengan jumlah lebih dari satu**

Pembuatan menu di dalam Visual Basic .NET tidak lagi menempatkan menu sebagai suatu fasilitas tersendiri sebagai sebuah menu editor, karena menu di dalam Visual Basic .NET merupakan sebuah control tersendiri. Sehingga mengakibatkan sebuah form dapat memiliki beberapa macam menu sekaligus. Selain itu, karena menu dalam Visual Basic .NET juga merupakan sebuah class, maka seorang programmer dapat melakukan penurunan sifat untuk membuat class baru berdasarkan control menu yang sudah ada.

Secara umum, di dalam sebuah bahasa pemrograman, khususnya yang berbasis visual, terdapat dua macam menu yang bisa dibuat programmer dalam sebuah form antara lain :

1. **Mainmenu atau menu pulldown**

   Yaitu menu yang terletak di bagian atas sebuah form. Menu jenis ini selalu muncul dalam aplikasi – aplikasi yang populer dan dianggap sebagai sebuah standard baku dari sebuah aplikasi berbasis Windows.

2. **Popup menu atau context menu atau shortcut menu**

   Merupakan menu yang biasanya muncul saat pengguna melakukan aksi klik kanan di sebuah bagian form. Penggunaan menu jenis ini secara umum hanya mengambil prosedur yang paling sering dipakai oleh pengguna dalam sebuah aplikasi untuk menghemat waktu. Prosedur yang ada dalam sebuah context menu seringkali juga terdapat di dalam mainmenu. Dalam interaksi manusia dengan komputer, khususnya bagi para pengguna pemula, menu jenis ini jarang sekali digunakan.

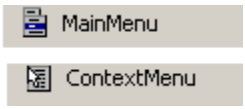
Sedangkan dalam proses pembuatannya, sebuah menu pulldown dibagi menjadi dua bagian utama yaitu :

1. **Menu utama**

   Merupakan bagian menu yang terlihat langsung oleh pengguna pertama kali. Bagian menu ini umumnya menerima shortcut dengan kombinasi tombol Alt dan huruf yang digarisbawah dalam teks yang tertera (dalam pemrograman, huruf yang diberi garis bawah, didepannya diberi tanda ampersand (&)). Sebuah menu utama dapat terdiri dari beberapa sub menu atau bahkan tidak memiliki sub menu sama sekali. Sedangkan di dalam Visual Basic .NET

sebuah menu utama tidak mampu menggunakan property *checked* yang menunjukkan status tanda cek þ di bagian kanan menu tersebut seperti halnya yang bisa terjadi di sub menu. Sebuah menu utama juga berfungsi sebagai induk dari sub menu yang ada dibawahnya, misal : jika sebuah menu utama diset property visiblenya menjadi false, maka seluruh sub menu yang ada di bawah menu utama tersebut, property visiblenya juga akan menjadi false secara otomatis.

2. **Sub menu atau menu item**
   Merupakan bagian dari menu utama dan secara hirarkis, terletak di bawah menu utama. Sebuah sub menu dapat memiliki sub menu yang lain.



Gambar 7.1 Control Mainmenu dan Contextmenu di Visual Basic .NET

Di dalam Visual Basic .NET , sebuah menu (baik mainmenu ataupun contextmenu) juga dapat dibuat secara dinamis. Dalam sebuah aplikasi, pembuatan menu secara dinamis seringkali kita temui, misal di dalam aplikasi Microsoft Office, sebuah menu dinamis bisa kita temui di bagian menu *File*, yaitu di bagian menu yang menampilkan dokumen – dokumen terakhir yang kita buka sebelumnya.

Jika kita memiliki beberapa control mainmenu yang terdapat dalam sebuah form sekaligus, maka untuk mengarahkan form ke mainmenu tertentu, dapat digunakan property mainmenu yang terdapat dalam form, baik pada saat design time ataupun pada saat run time.

Beberapa property dari menu yang sering digunakan yaitu :

1. **Text**
2. **Checked**
3. **DefaultItem**
   Untuk menandakan sebuah bagian menu sebagai menu default dalam sebuah aplikasi. Pada saat program dieksekusi, bagian menu yang property defaultitemnya diset menjadi true akan dicetak tebal untuk membedakan dengan bagian menu yang lain.

4. **Shortcut**

   Untuk mengarahkan jalan pintas kombinasi penekanan tombol keyboard untuk menuju ke bagian menu tertentu. Usahakan tidak ada shortcut yang sama dalam suatu menu.
5. **Visible**
6. **Enabled**

Soal latihan :

Buat sebuah form dan tempatkan sebuah menu dalam form tersebut dengan struktur sebagai berikut :

Gambar 7.2 Struktur menu soal latihan

Jawab :

1. **Buat form dan drag sebuah control mainmenu dari toolbox**
2. **Buat dua bagian menu utama**
3. **Di bawah bagian menu utama *Test Menu* tambahkan dua sub menu seperti pada gambar struktur menu.**
4. **Klik ganda sub menu *Pesan 1* untuk mengisikan perintah yang akan dilakukan di sub menu tersebut, ketikkan listing untuk menampilkan pesan berikut :**

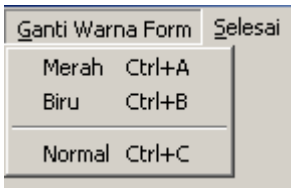
   ```
   MessageBox.Show("Test pesan 1")
   ```
5. **Lakukan hal yang sama di sub menu *Pesan 2***

   ```
   MessageBox.Show("Test pesan 2")
   ```
6. **Sedangkan untuk bagian menu utama *Selesaii* yang tidak memiliki sub menu, juga dapat dilakukan hal yang sama seperti pada sub menu *Pesan 1* ataupun *Pesan 2***
7. **Eksekusi program untuk melihat hasilnya.**

Jika hanya ada satu control mainmenu dalam form, maka secara otomatis property mainmenu dalam form akan mengarah ke control tersebut.

Soal latihan :

Buat form dengan struktur menu seperti pada gambar, dan berikan tanda cek secara toggle (hanya berlaku sekali untuk sebuah sub menu) jika sebuah menu sedang dalam keadaan dipilih.



Gambar 7.3 Struktur menu dengan shortcut dan separator

Jawab :

1. **Buat form dan drag sebuah control mainmenu**
2. **Atur property di tiap sub menu untuk shortcut masing – masing.**
3. **Pada sub menu *Biru*, lakukan klik kanan hingga muncul menu bari dan pilih sub menu *Insert Separator* untuk memunculkan garis pemisah antara sub menu tersebut dengan sub menu *Normal***
4. **Buat sebuah prosedur untuk menangani pemilihan form sekaligus mengatur agar tanda cek hanya terlihat di sub menu yang sedang aktif :**

```
Private Sub cariMenuItem (ByVal x As MenuItem, _
    ByVal y As Color)
    For Each i As MenuItem In MenuItem1.MenuItems
        If i.Checked = True Then i.Checked = False
    Next
    x.Checked = True
    Me.BackColor = y
End Sub
```

5. **Pada tiap sub menu panggil prosedur yang ada untuk mengganti warna dari form :**

   a. **Merah**

   ```
   cariMenuItem(sender, Color.Red)
   ```

   b. **Biru**

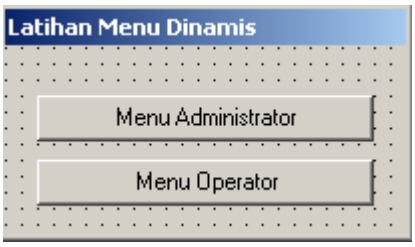
   ```
   cariMenuItem(sender, Color.Blue)
   ```

   c. **Normal**

   ```
   cariMenuItem(sender, Color.Empty)
   ```

Soal latihan :

Buat sebuah form dengan dua buah tombol, aplikasi pembuatan menu dinamis untuk masing – masing menu berdasarkan dari tombol yang diklik oleh pengguna. Buat prosedur yang akan menangani pemilihan sub menu dari menu dinamis yang akan dibuat.



Gambar 7.4 Layout form untuk menu dinamis

Jawab :

1. **Buat sebuah form seperti pada gambar**
2. **Buat sebuah prosedur untuk menangani menu selesai**

```
Private Sub selesai (ByVal sender As Object, _
    ByVal e As System.EventArgs)
    Dispose()
End Sub
```

3. **Buat prosedur untuk menangani penekanan sub menu**

```
Private Sub tampilPesan (ByVal sender As Object, _
    ByVal e As System.EventArgs)
    MessageBox.Show(sender.text)
End Sub
```

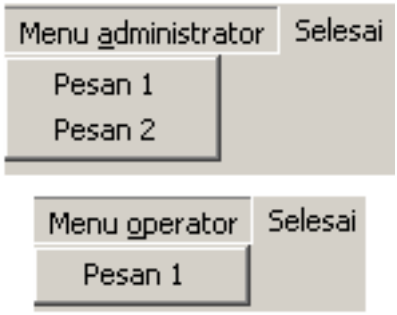
4. **Di dalam button *Menu Administrator* ketikkan listing berikut :**

```
Dim menuAdmin As New MainMenu
Dim menuUtama1 As New MenuItem
Dim menuUtama2 As New MenuItem
Dim subMenu1 As New MenuItem
Dim subMenu2 As New MenuItem
'tambah bagian menu utama
menuAdmin.MenuItems.Add(menuUtama1)
menuAdmin.MenuItems.Add(menuUtama2)
'tambah sub menu
menuUtama1.MenuItems.Add(subMenu1)
menuUtama1.MenuItems.Add(subMenu2)
'set property bagian menu
menuUtama1.Text = _
        "Menu &administrator"
menuUtama2.Text = "Selesai"
subMenu1.Text = "Pesan 1"
subMenu2.Text = "Pesan 2"
'tambah prosedur dinamis
AddHandler menuUtama2.Click, AddressOf selesai
```

```
AddHandler subMenu1.Click, AddressOf tampilPesan
AddHandler subMenu2.Click, AddressOf tampilPesan
'set property form
Menu = menuAdmin
```

5. Di dalam button *Menu Operator* ketikkan listing berikut :

```
Dim menuOperator As New MainMenu
Dim menuUtama1 As New MenuItem
Dim menuUtama2 As New MenuItem
Dim subMenu1 As New MenuItem
Dim subMenu2 As New MenuItem
'tambah bagian menu utama
menuOperator.MenuItems.Add(menuUtama1)
menuOperator.MenuItems.Add(menuUtama2)
'tambah sub menu
menuUtama1.MenuItems.Add(subMenu1)
'set property bagian menu
menuUtama1.Text = "Menu &operator"
menuUtama2.Text = "Selesai"
subMenu1.Text = "Pesan 1"
'tambah prosedur dinamis
AddHandler menuUtama2.Click, AddressOf selesai
AddHandler subMenu1.Click, AddressOf tampilPesan
'set property form
Menu = menuOperator
```



Gambar 7.5 Hasil struktur menu

þ Keterangan

Teknik pembuatan menu secara dinamis juga bisa dilakukan kepada control yang lain, yaitu dengan melakukan deklarasi obyek (dengan ciri adanya kata kunci *New*)dengan tipe yang sesuai. Sedangkan untuk prosedur yang akan menangani control yang dibuat secara dinamis, terlebih dahulu didefinisikan dengan melakukan *overriding* (akan dijelaskan lebih lanjut di bab *pemrograman berorientasi obyek*) terhadap method dari control yang bersangkutan. Artinya bahwa prosedur yang akan menangani juga memiliki parameter standard dari method aslinya (dengan parameter sender dan e).
Penggunaan perintah *AddHandler* digunakan untuk mengarahkan sebuah prosedur yang sebelumnya telah ada ke sebuah control yang dibuat secara dinamis pada saat run time.

Sintaks perintah tersebut adalah *AddHandler (nama method dari control), AddressOf (nama prosedur yang akan didelegasikan)*.
Pembuatan menu membutuhkan perancangan tentang struktur menu yang akan dibuat. Tentukan bagian utama dari menu serta sub menu dari masing – masing menu utama. Untuk bagian menu utama ataupun sub menu memiliki tipe yang sama yaitu *menutitem*.
Selain itu, juga harus didefinisikan property tiap sub menu yang ada, baik property *text* ataupun property yang lain seperti *shortcut* dan *checked*.

Sebuah context menu dapat diarahkan bukan hanya kepada form, tetapi juga ke tiap control yang ada dalam form, ataupun di tiap koordinat yang berbeda. Dengan kata lain, context menu dapat bersifat *context sensitive* atau berbeda di tiap bagian form.

Jika sebuah control contextmenu ditempatkan dalam sebuah form, maka secara otomatis, di tiap control yang ada di form tersebut property contextmenunya akan langsung dapat diarahkan ke control contextmenu tersebut saat design time.

Soal latihan :

Buat sebuah form kosong, tempatkan sebuah contextmenu dengan struktur seperti pada gambar berikut :

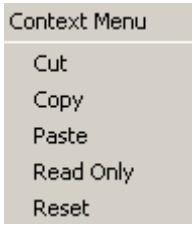
Gambar 7.6 Struktur contextmenu

Jawab :

1. **Buat form dan drag control contextmenu, lalu set struktur menu seperti pada gambar**
2. **Klik ganda pada sub menu *Ganti Merah* dan ketikkan listing berikut :**
   ```
   Me.BackColor = Color.Red
   ```
3. **Klik ganda pada sub menu *Ganti Normal* dan ketikkan listing berikut :**
   ```
   Me.BackColor = Color.Empty
   ```
4. **Klik ganda pada sub menu *Selesai* dan ketikkan listing berikut :**
   ```
   Close
   ```
5. **Set property contextmenu di form agar mengarah ke contextmenu yang baru dibuat.**
6. **Eksekusi program dan lakukan klik kanan pada form**

Soal latihan :

Buat form dengan sebuah textbox didalamnya, dan buat contextmenu khusus untuk textbox tersebut. Contextmenu akan berlaku dinamis sesuai dengan kondisi dan isi dari textbox tersebut.



Gambar 7.7 Struktur contextmenu dinamis

Jawab :

1. **Buat sebuah form dan tempatkan sebuah textbox serta sebuah contextmenu.**
2. **Arahkan property contextmenu textbox ke control contextmenu yang baru ditempatkan**
3. **Atur struktur contextmenu seperti pada gambar**
4. **Di dalam sub menu *Cut* ketikkan listing berikut :**

```
If TextBox1.Text <> "" Then
    TextBox1.SelectAll()
    TextBox1.Cut()
End If
```

5. **Di dalam sub menu *Copy* ketikkan listing berikut :**

```
If TextBox1.Text <> "" Then
    TextBox1.SelectAll()
    TextBox1.Copy()
End If
```

6. **Di dalam sub menu *Paste* ketikkan listing berikut :**

```
Dim iData As IDataObject = Clipboard.GetDataObject()
If CType(iData.GetData (DataFormats.Text), String) _
    <> "" Then
    TextBox1.Text = ""
    TextBox1.Paste()
End If
```

þ Keterangan

Di dalam Visual Basic .NET , setiap operasi clipboard (cut, copy dan paste) selalu menghasilkan data bertipe object. Hal ini dikarenakan operasi clipboard juga bisa dihasilkan dari penekanan tombol *PrintScreen* di keyboard yang akan menghasilkan data bertipe image hasil capture layar yang sedang aktif. Karenanya, programmer harus berhati – hati dalam menggunakan hasil operasi clipboard.

Dalam listing tersebut, hasil operasi clipboard berusaha ditangkap dengan menggunakan method *GetDataObject* dari class *Clipboard* yang terdapat dalam Visual Basic .NET , kemudian berusaha dikonversi ke dalam tipe string dengan fungsi *Ctype*.

7. **Di dalam sub menu *Read Only* ketikkan listing berikut :**

```
If TextBox1.ReadOnly = False Then
    TextBox1.ReadOnly = True
    sender.checked = True
Else
    TextBox1.ReadOnly = False
    sender.checked = False
End If
```

8. **Di dalam sub menu *Reset* ketikkan listing berikut :**

```
TextBox1.Text = ""
```

9. **Klik ganda pada contextmenu, dan di dalam method *PopUp* ketikkan listing berikut untuk mendeteksi agar sub menu yang muncul bisa menyesuaikan dengan keadaan textbox.**

```
If TextBox1.Text = "" Then
          'disable sub menu cut
          MenuItem1.Enabled = False
    'disable sub menu copy
    MenuItem2.Enabled = False
    'disable sub menu reset
    MenuItem5.Enabled = False
Else
    'enable semua menu
    For i As Integer = 0 To _
        sender.MenuItems.Count - 1
        sender.MenuItems.Item(i).Enabled = True
    Next
End If
```

10. **Eksekusi program dan cek program dengan mengisi textbox dan melakukan klik kanan, juga cek jika textbox dalam keadaan kosong.**

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 yaitu :

1. **Single Document Interface**
2. **Multiple Document Interface**

Single 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**

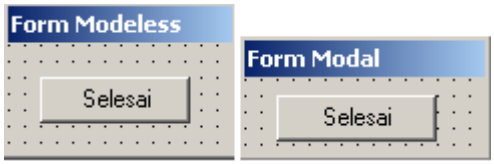
   Mengijinkan pengguna untuk dapat berpindah antar form secara dinamis.
2. **Form Modal**

   Mengasumsikan bahwa form majemuk memiliki hirarki tertentu, sehingga form yang lain diasumsikan tidak bisa diproses sebelum form yang sedang aktif sudah selesai ditutup.

Soal latihan :

Buat tiga buah form dalam sebuah solution, dengan layout seperti pada gambar berikut :

Gambar 7.8 Layout dan struktur menu Form1



Gambar 7.9 Layout Form2 dan Form3

Jawab :

1. **Buat sebuah solution baru dengan tiga buah form yang layoutnya seperti pada gambar**
2. **Pada form pertama tempatkan satu mainmenu dengan struktur menu seperti di gambar**
3. **Di dalam sub menu *Form Kedua* ketikkan listing berikut :**

```
Dim frm2 As New Form2
frm2.Show()
```

4. **Di dalam sub menu *Form Ketiga* ketikkan listing berikut :**

```
Dim frm3 As New Form3
frm3.ShowDialog()
```

5. **Di dalam sub menu *Form Kedua Kuning* ketikkan listing berikut :**

```
Dim frm2 As New Form2
frm2.Show()
frm2.BackColor = Color.Yellow
```

6. **Di dalam sub menu *Form Ketiga Merah* ketikkan listing berikut :**

```
Dim frm3 As New Form3
frm3.BackColor = Color.Red
frm3.ShowDialog()
```

7. **Eksekusi program dan perhatikan perbedaan antara form kedua dan form ketiga. Form kedua menggunakan teknik form Modeless, sedangkan form ketiga menggunakan teknik form Modal.**

Multiple Document Interface

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.

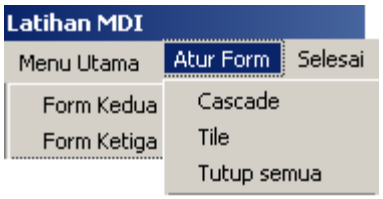
Teknik MDI akan menyebabkan semua form yang dianggap sebagai child akan selalu berada di dalam form induk atau form parent. Banyak sekali aplikasi yang menggunakan teknik MDI, misal : Microsoft Office. Dalam teknik MDI,

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).

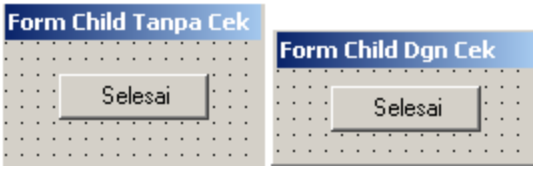
Sebuah aplikasi form majemuk dengan menggunakan teknik MDI, memiliki satu form parent dan minimal satu form sebagai child. Sebuah form parent ditandai dengan adanya nilai true pada property *IsMDIContainer*. Saat form parent ditutup, maka secara otomatis seluruh form child yang sedang terbuka ikut tertutup.

Soal latihan :

Buat sebuah aplikasi yang memiliki tiga buah form dengan satu form utama sebagai *parent*.



Gambar 7.10 Layout form utama MDI dan struktur menu



Gambar 7.11 Layout Form2 dan Form3

Jawab :

1. **Buat solution baru dengan tiga buah form yang memiliki layout seperti pada gambar.**
2. **Pada form pertama, tempatkan sebuah control mainmenu dengan struktur seperti pada gambar**
3. **Pada form pertama, set property *IsMDIContainer* menjadi true**
4. **Pada sub menu *Form Kedua* ketikkan listing berikut :**

```
Dim frm2 As New Form2
frm2.MdiParent = Me
frm2.Show()
```

þ Keterangan

Untuk mengarahkan form sebagai form child dari form pertama, maka digunakan property *MdiParent* dengan memberi nilai ke form yang sedang aktif, yaitu form yang dianggap sebagai form parent. Property ini hanya berlaku saat run time.

5. Pada sub menu *Form Ketiga* ketikkan listing berikut :

```
Dim frm3 As New Form3
For Each i As Form In Me.MdiChildren
    If i.Name = "Form3" Then i.Close()
Next
frm3.MdiParent = Me
frm3.Show()
```

þ Keterangan

Untuk form ketiga, dilakukan pengecekan terlebih dahulu agar tidak terjadi pemunculan form yang sama lebih dari satu kali.

6. Pada sub menu *Cascade* ketikkan listing berikut :

```
LayoutMdi(MdiLayout.Cascade)
```

7. Pada sub menu *Tile* ketikkan listing berikut :

```
LayoutMdi(MdiLayout.TileHorizontal)
```

8. Pada sub menu *Tutup Semua* ketikkan listing berikut :

```
For Each i As Form In MdiChildren
    i.Close()
Next
```

9. Eksekusi program dan bedakan penggunaan sub menu untuk form kedua dan ketiga.

Visual Basic .NET telah menyederhanakan pembuatan aplikasi form majemuk dengan teknik MDI. Hal ini dengan adanya property baru *IsMDIContainer* dan mengeliminasi penggunaan jenis form khusus untuk membuat sebuah form parent. Akibatnya, form parent dapat berganti secara dinamis dalam suatu aplikasi MDI.

Penanganan Kesalahan

Tujuan :

- **Memahami dasar penanganan kesalahan dalam pembuatan aplikasi**
- **Mampu membuat sendiri penanganan kesalahan dalam pengembangan aplikasi menggunakan Visual Basic .NET**

Error Provider

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.

Gambar 8.1 Error provider dalam toolbox

Beberapa property yang sering digunakan dalam penggunaan error provider adalah :

1. **BlinkRate**

   Mengatur kecepatan kedip error provider saat terjadi kesalahan.

2. **Icon**

   Untuk mengganti icon default dari error provider.

Error provider juga sering digunakan sebagai penanda dalam suatu control yang lain, misal : datagrid dalam suatu aplikasi database.

Teknik pemrograman untuk sebuah penanganan kesalahan atau error handling sendiri secara garis besar dibagi menjadi dua yaitu :

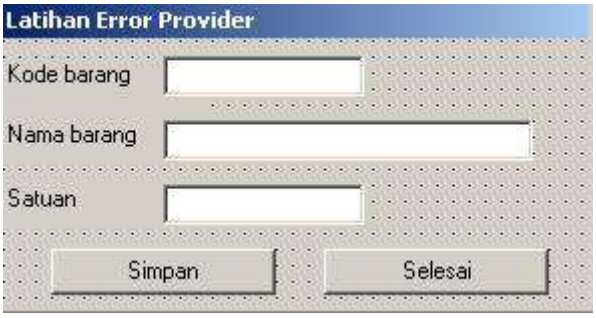
1. **Field Validation**
2. **Form Validation**

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.

Soal latihan :

Buat sebuah form untuk mendemonstrasikan penggunaan error provider. Gunakan teknik field validation untuk melakukan pengecekan di tiap textbox agar tidak ada textbox yang tidak terisi.



Gambar 8.2 Layout form latihan error

Jawab :

1. **Buat form seperti pada gambar dan tempatkan sebuah control error provider (tidak tampak dalam form)**

2. **Isikan di property tag tiap textbox dengan teks seperti yang tertera pada keterangan label, misal untuk textbox1, isi dari property tag adalah *kode barang***
3. **Klik ganda pada textbox pertama, kemudian pindahkan ke method *validating*, lalu ketikkan listing berikut :**

```
Private Sub TextBox1_Validating (ByVal sender As Object, _
    ByVal e As System.ComponentModel.CancelEventArgs) _
    Handles TextBox1.Validating, TextBox2.Validating, _
    TextBox3.Validating
    If sender.text = "" Then
        ErrorProvider1.SetError(sender, sender.tag & _
            " tidak boleh kosong !")
        MessageBox.Show(sender.tag & " tidak boleh kosong !", _
            "Salah")
        e.Cancel = True
    End If
End Sub
```

þ Keterangan

Untuk menampilkan error provider sebagai tanda adanya kesalahan, digunakan method *SetError* untuk menampilkan error provider ke dalam form. Tetapi kebanyakan pengguna kurang fokus terhadap error provider karena ukuran iconnya yang terlalu kecil serta pesan kesalahan yang hanya muncul jika kursor diarahkan ke atas area error provider. Sehingga seringkali tetap digunakan perintah messagebox untuk menarik perhatian pengguna tentang adanya kesalahan dalam proses. Pada baris *e.cancel = true* akan mengakibatkan kursor tetap kembali ke dalam textbox sebelum diisikan nilai yang valid.

Hati – hatilah dalam menggunakan validasi, seringkali validasi menyebabkan pengguna merasa frustrasi dalam proses pengisian data, karena merasa tidak pernah bisa keluar dari keadaan yang salah.

4. **Klik ganda lagi pada textbox pertama, tetapi kali ini pindahkan ke method *validated*, lalu ketikkan listing berikut ini :**

```
Private Sub TextBox1_Validated _
    (ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles TextBox1.Validated, TextBox2.Validated, _
    TextBox3.Validated
    ErrorProvider1.SetError(sender, "")
End Sub
```

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.

Soal latihan :

Buat form seperti pada latihan soal sebelumnya, tapi kali ini dengan menggunakan teknik form validation.

Jawab :

1. **Lakukan langkah yang sama seperti soal sebelumnya, tetapi pada method validating dan validated tidak perlu diisi.**
2. **Klik ganda pada tombol *Simpan*, lalu ketikkan listing berikut ini :**

```
Dim xlolos As Boolean = True
For Each i As Object  In Me.Controls
    ErrorProvider1.SetError (i, "")
Next
For Each i As Object In Me.Controls
    If TypeOf i Is TextBox Then
        If i.text = "" Then
            ErrorProvider1.SetError _
            (i, i.tag & " tidak boleh kosong !")
            i.Focus()
            xlolos = False
        End If
    End If
Next
If xlolos Then
    MessageBox.Show("Lolos !")
    'tempatkan proses asli
End If
```

þ Keterangan

Dalam form validation, error provider harus dibersihkan terlebih dulu dari pesan kesalahan. Hal ini untuk mencegah *tertinggalnya* pesan kesalahan saat terjadi proses yang kedua atau ketiga, sedangkan proses – proses tersebut telah terhindar dari kesalahan.

3. Eksekusi program dan perhatikan perbedaan dari sisi pengguna antara field validation dan form validation.

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.

Soal latihan :

Buat form untuk kalkulator sederhana, tetapi kali ini dengan pengecekan apakah inputan dari pengguna bisa dilakukan operasi matematika, serta juga mencegah kesalahan jika terjadi kesalahan *overflow* dalam perhitungan matematika, misal : jika terjadi perhitungan angka satu dibagi dengan angka nol (menjadi tak terhingga)

Gambar 8.3 Layout form latihan error

Jawab :

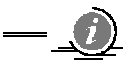
1. **Buat form seperti pada gambar**
2. **Di dalam button untuk penjumlahan, ketikkan listing berikut :**

```
Private Sub Button1_Click _
  (ByVal sender As System.Object, ByVal e As System.EventArgs) _
  Handles Button1.Click, Button2.Click, Button3.Click, _
  Button4.Click
    Dim hasil As Integer = 0
    Dim bil1 As Integer
    Dim bil2 As Integer
    Try
        bil1 = TextBox1.Text
        bil2 = TextBox2.Text
        Try
            Select Case sender.text
                Case "+"
                    hasil = bil1 + bil2
                Case "-"
                    hasil = bil1 - bil2
                Case "/"
                    hasil = bil1 / bil2
                Case "x"
                    hasil = bil1 * bil2
            End Select
            MessageBox.Show(hasil)
        Catch ex As Exception
           MessageBox.Show("Hasil salah !")
        End Try
    Catch ex As Exception
        MessageBox.Show("Inputan salah !")
    End Try
End Sub
```

þ Keterangan

Try...Catch yang pertama melakukan pengecekan terhadap nilai yang terdapat dalam textbox, sedangkan yang kedua bertugas untuk melakukan penanganan kesalahan jika terjadi nilai ilegal pada hasil operasi matematika.

3. **Eksekusi program dengan mencoba memasukkan huruf (bukan angka) ke dalam tiap textbox. Kemudia tes juga dengan memasukkan angka satu di textbox pertama dan angka nol di textbox kedua, lalu lakukan operasi pembagian.**



Dalam pemrograman tingkat lanjut, programmer dapat membuat sebuah class baru untuk menangani exception baru yang khusus dipergunakan untuk sebuah aplikasi tertentu.

Komponen Standard Lain

Tujuan :

- **Memahami penggunaan komponen standard selain yang telah dibahas di bab sebelumnya**
- **Mampu mengaplikasikan komponen standard lain ke dalam aplikasi dengan dalam Visual Basic .NET**

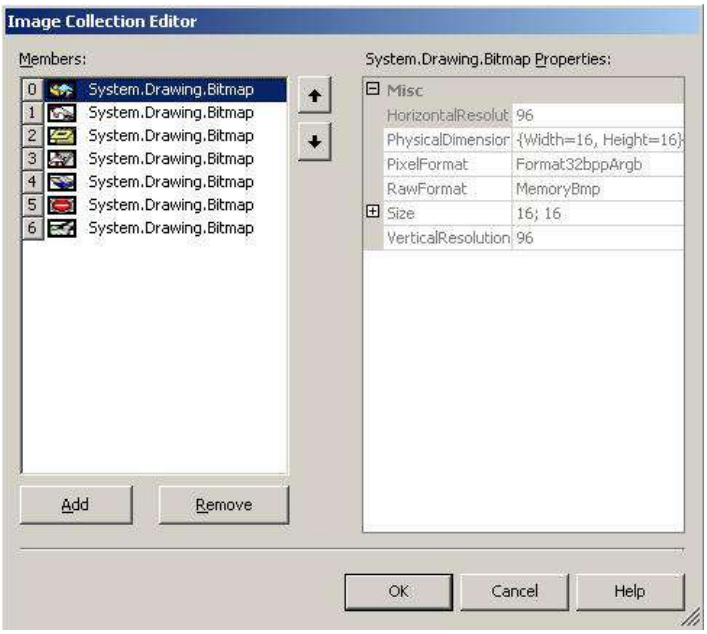
Imagelist

Imagelist adalah control yang mampu menampung berbagai format data image dalam suatu collection, dan dapat dipergunakan untuk control yang lain, misal : treeview, listview ataupun toolbar.

Gambar 9.1 Imagelist dalam toolbox

Sebuah imagelist dapat diisi file image di dalam run time ataupun di dalam design time. Property yang digunakan untuk menampung file image tersebut adalah *images*. Untuk mengakses property tersebut ke control yang lain dapat digunakan indeks dari collection yang sudah ada dalam imagelist tersebut.

Programmer seringkali beranggapan bahwa imagelist hanya mampu menampung image berukuran kecil untuk icon, padahal imagelist mampu menampung image dengan ukuran yang bervariasi.



Gambar 9.2 Kotak dialog collection di dalam imagelist

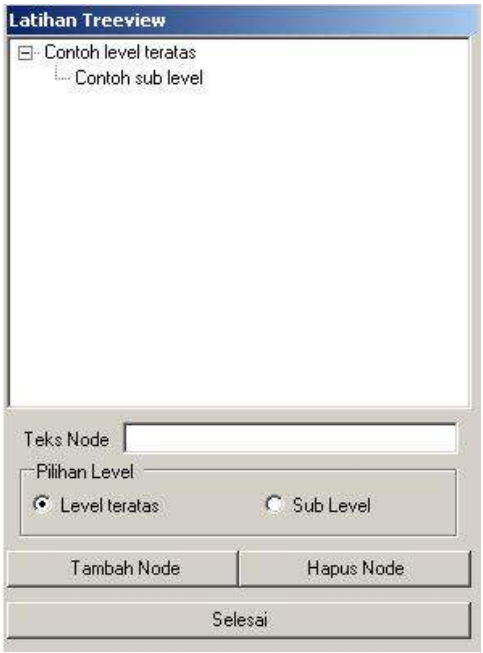
Treeview

Merupakan control atau komponen standard lain yang ada di dalam Visual Basic .NET untuk menampilkan data atau array secara hirarki dalam tiap treenode-nya. Sebuah treeview minimal terdiri dari dua level yaitu *root node* atau level yang lebih tinggi dan *child node* atau level yang merupakan sub level dari *root node*. Sebuah *child node* dapat menjadi sebuah *root node* jika didalamnya terdapat *child node* yang lain.

Gambar 9.3 Treeview dalam toolbox

Soal latihan :

Buat program untuk mendemonstrasikan penambahan dan pengurangan node di dalam treeview secara interaktif dari pihak pengguna (dalam aplikasi, treeview seringkali digunakan untuk menampilkan struktur suatu direktori atau bahkan struktur skema dari suatu file XML). Layout form seperti pada gambar berikut :



Gambar 9.4 Layout form soal treeview

Jawab :

1. **Buat form seperti pada gambar dengan menggunakan treeview**
2. **Di dalam *form_load* isikan node awal dalam treeview**

```
Dim newNode As TreeNode = New TreeNode("Contoh sub level")
TreeView1.Nodes.Add ("Contoh level teratas")
TreeView1.Nodes(0).Nodes.Add(newNode)
```

3. **Di dalam tombol *Tambah node* isikan listing berikut :**

```
If RadioButton1.Checked Then
    TreeView1.Nodes.Add(TextBox1.Text)
Else
    Try
        Dim newNode As TreeNode = New TreeNode _
          (TextBox1.Text)
        TreeView1.Nodes (TreeView1.SelectedNode. _
            Index).Nodes.Add(newNode)
    Catch ex As Exception
        MessageBox.Show ("Node gagal !")
    End Try
End If
```

4. **Di dalam tombol *Hapus node* isikan listing berikut :**

```
Try
        TreeView1.Nodes.Remove(TreeView1.SelectedNode)
   Catch ex As Exception
        MessageBox.Show("Node gagal dihapus !")
   End Try
```

Listview

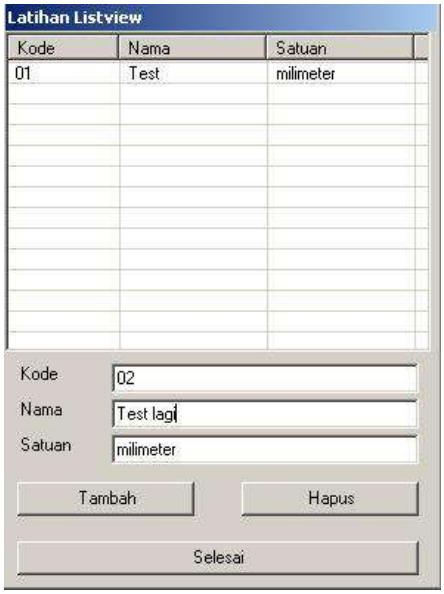
Control listview ada dasarnya adalah sebuah listbox yang mampu menampilkan grid didalamnya serta mampu menampung berbagai tipe data selain teks, misal tipe bitmap, selain itu listview juga mampu menampilkan data lebih dari satu kolom dibandingkan dengan listbox.

Gambar 9.5 Listview di dalam toolbox

Sebuah listview memiliki property *columns* yang merupakan collection dari kolom – kolom yang akan ditampilkan dalam listview tersebut. Sedangkan item – item yang ditampilkan di dalam kolom tersebut diatur dalam property *listviewitem* yang masing – masing memiliki collection *subitem* yang berfungsi sebagai isi teks dari tiap koordinat baris dan kolom tersebut.

Soal latihan :

Buatlah sebuah program untuk mengisikan data dalam sebuah listview yang mampu menampung data secara interaktif dari pengguna (dalam sebuah aplikasi database, listview dapat dijadikan sebagai pengganti datagrid dalam kasus tertentu).



Gambar 9.6 Soal latihan listview

Jawab :

1. **Setelah membuat form seperti pada gambar, isikan listing berikut di dalam *form_load***

```
ListView1.View = View.Details
ListView1.Columns.Add ("Kode", 75, _
    HorizontalAlignment.Center)
ListView1.Columns.Add ("Nama", 100, _
    HorizontalAlignment.Left)
ListView1.Columns.Add ("Satuan", 100, _
    HorizontalAlignment.Left)
```

2. **Di dalam button *Tambah* ketikkan listing berikut :**

```
Dim xList As New ListViewItem(TextBox1.Text)
       xList.SubItems.Add(TextBox2.Text)
       xList.SubItems.Add(TextBox3.Text)
       ListView1.Items.Add(xList)
```

3. **Di dalam button *Hapus* ketikkan listing berikut :**

```
Try
    ListView1.Items.Remove (ListView1.FocusedItem)
Catch ex As Exception
    MessageBox.Show("Gagal dihapus !")
End Try
```

þ Keterangan

Dalam pembuatan listview, tentukan dulu kolom yang akan ditampilkan serta tipe dari listview itu sendiri (grid, normal atau menggunakan image dengan imagelist). Kemudian inisialisasi obyek listviewitem sebelum mengisikan data di dalam listview tersebut.

Progressbar

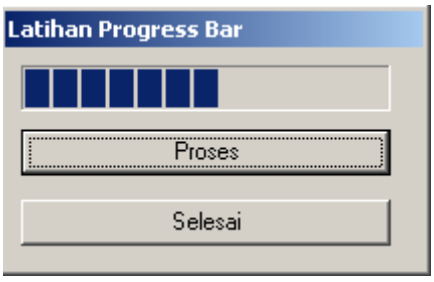
Progressbar adalah control yang akan menampilkan hasil kemajuan dari suatu proses tertentu dengan menampilkan kotak – kotak kecil dalam sebuah container khusus.

Gambar 9.7 Progressbar di dalam toolbox

Secara default, sebuah progress bar akan menunjukkan skala nilai minimum – maksimum antara 0 sampai dengan 100 dalam menyatakan kemajuan suatu proses. Tetapi skala tersebut dapat diganti dengan menggunakan property *minimum* dan *maximum*. Sedangkan untuk menyatakan operasi kemajuan proses, bisa digunakan method *performStep* atau dengan menambahkan property *value* dengan pencacah yang dimaksud.

Soal latihan :

Buat aplikasi untuk mendemonstrasikan penggunaan progressbar dengan menggunakan perintah perulangan.



Gambar 9.8 Soal latihan progressbar

Jawab :

1. **Buat form dengan layout seperti pada gambar**
2. **Di dalam button *Proses* ketikkan listing berikut :**

```
ProgressBar1.Value = 0
For i As Integer = 1 To 100000000
```

```
    If i Mod 1000000 = 0 Then _
     ProgressBar1.PerformStep()
Next
```

Timer

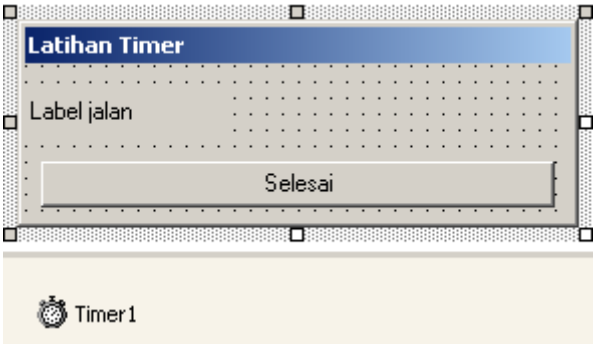
Timer adalah control yang dapat membangkitkan event tertentu sesuai dengan interval waktu yang telah ditentukan dalam program. Interval waktu timer menggunakan satuan milidetik.

Gambar 9.9 Timer di dalam toolbox

Timer merupakan control yang tidak tampak saat design time maupun pada saat run time. Besarnya interval waktu dapat diset di dalam property *interval*, dan interval akan berjalan hanya jika property enabled dari timer itu diset menjadi *true*.

Soal latihan :

Buat form yang dapat menampilkan sebuah label yang berjalan dari kiri ke kanan secara simultan (seperti teknik marquee di dalam pemrograman web) dengan menggunakan timer.



Gambar 9.10 Layout soal timer

Jawab :

1. **Buat form seperti pada gambar**
2. **Set property timer sebagai berikut :**

Enabled	**True**
Interval	**50**

Gambar 9.11 Setting property timer

3. **Klik ganda pada timer, lalu ketikkan listing berikut :**

```
If Label1.Left >= Me.Width Then
    Label1.Left = -Label1.Left
End If
Label1.Left += 10
```

Semakin kecil nilai property interval akan menyebabkan label berjalan lebih cepat dan sebaliknya.

Soal latihan :

Buatlah sebuah form yang dapat menampilkan karakter (*) sebagai bintang yang mampu berkedip dan bertebaran di dalam sebuah form.

Gambar 9.12 Layout soal timer kedua

Jawab :

1. **Buat layout form seperti pada gambar**
2. **Set property *KeyPreview* pada form menjadi *true***
3. **Set property *interval* pada timer menjadi 50 dan property enablednya menjadi *true***
4. **Klik ganda pada timer dan ketikkan listing berikut :**

```
Randomize()
Dim x As Integer = Int((Rnd() * Me.Width) + 1)
Dim y As Integer = Int((Rnd() * Me.Height) + 1)
Label1.Left = x
Label1.Top = y
Dim r As Integer = Int(Rnd() * 255)
```

```
Dim g As Integer = Int(Rnd() * 255)
Dim b As Integer = Int(Rnd() * 255)
Label1.ForeColor = Color.FromArgb(100, r, g, b)
```

5. Di dalam *form _keyup* ketikkan listing berikut :

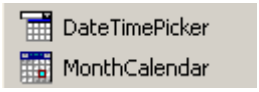
```
If e.KeyCode = Keys.F2 Then Close()
```

þ Keterangan

Perintah *Randomize* akan membangkitkan nilai acak yang selanjutnya dapat diambil melalui fungsi *rnd*. Fungsi *rnd* akan mengambil sebuah nilai acak dari angka 0 hingga 1. Agar angka acak dapat lebih besar dari 1, maka kalikan hasil fungsi *rnd* dengan nilai batas atas dan tambahkan dengan nilai batas bawah (lihat di dalam listing program).
Fungsi color *ARGB* akan melakukan pengambilan warna dengan parameter *alpha* untuk transpransi (bernilai 100 jika tidak transparan, dan bernilai 0 untuk transparan), *red* untuk kadar warna merah (dengan jangkauan angka antara 0 hingga 255), *green* untuk kadar warna hijau serta *blue* untuk kadar warna biru.

DateTimePicker

DateTimePicker pada dasarnya merupakan dua komponen yang mirip, yaitu untuk menampilkan dan mengisi data dengan format tanggal. Format tanggal yang tampil di dalam control tersebut, bergantung kepada format tanggal masing – masing sistem operasi yang ada pada pengguna. Untuk melakukan setting format tanggal dapat dilakukan di dalam Visual Basic .NET ataupun di dalam *regional setting* yang terdapat di dalam control panel.



Gambar 9.13 DateTimePicker dalam toolbox

Soal latihan :

Buat sebuah form yang dapat menghitung umur seseorang berdasarkan inputan tanggal lahir.

Gambar 9.14 Layout soal latihan DateTimePicker

Jawab :

1. **Buat form seperti pada gambar**
2. **Klik ganda pada tombol *Hitung umur* lalu ketikkan listing berikut :**

```
Dim xtahun As Integer
xtahun = DateDiff(DateInterval.Year, _
    DateTimePicker1.Value, Now)
MessageBox.Show("Umur anda : " & _
    xtahun & " tahun " )
```

þ Keterangan

Fungsi *DateDiff* digunakan untuk menghitung selisih antara dua nilai tanggal berdasarkan parameter yang telah ditentukan (dalam contoh parameternya adalah selisih tahun)

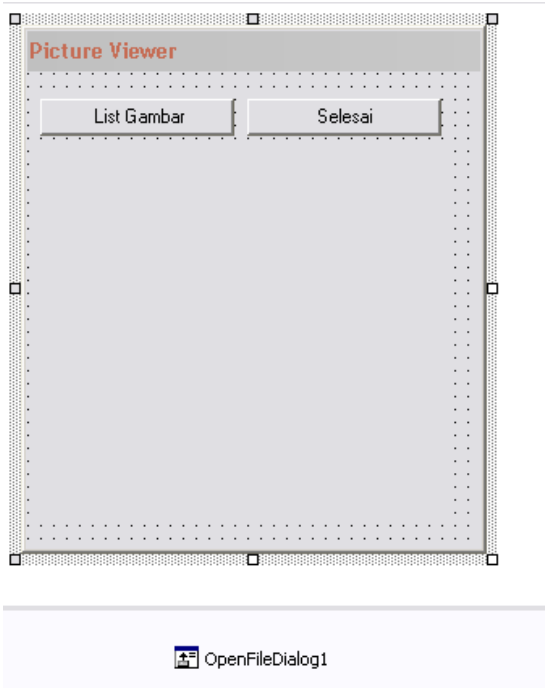
PictureBox

Picturebox merupakan control yang digunakan untuk menampilkan gambar dalam sebuah form, baik gambar yang berasal dari sebuah file ataupun gambar yang dihasilkan dari class GDI+ (Graphics Drawing Interface).

Gambar 9.15 PictureBox dalam toolbox

Soal latihan :

Buatlah sebuah form yang berfungsi untuk menampilkan gambar dari sebuah kotak dialog *OpenFile* dan menampilkannya dalam sebuah control PictureBox.



Gambar 9.16 Layout soal latihan PictureBox

Jawab :

1. **Buat form seperti pada gambar**
2. **Set property *sizeMode* pada PictureBox menjadi *StrecthImage***
3. **Klik ganda pada tombol *List Gambar* lalu ketikkan listing berikut :**

```
With OpenFileDialog1
    .Filter = "Gambar JPG " & "(*.jpg)|*.jpg|" & _
        "Gambar Bitmap (*.bmp)|*.bmp "
    .ShowDialog()
    Dim xgambar As _
     New Bitmap(.FileName)
    PictureBox1.Image = xgambar
End With
```

Property *StrectchImage* berfungsi untuk menampilkan gambar yang ukurannya akan disesuaikan dengan ukuran dari PictureBox, sehingga gambar akan terlihat utuh dan tidak terpotong, meski berukuran lebih besar dari PictureBox itu sendiri.

þ Keterangan

Control *OpenFileDialog* digunakan untuk menampilkan kotak dialog yang akan memilih dan membuka sebuah file dengan filter ekstensi tertentu.

Picturebox merupakan control yang digunakan untuk menampilkan gambar dalam sebuah form, baik gambar yang berasal dari sebuah file ataupun gambar yang dihasilkan dari class GDI+ (Graphics Drawing Interface).

Gambar 9.17 StatusBar dalam toolbox

Soal latihan :

Buat sebuah form untuk mendemonstrasikan penggunaan StatusBar dengan menempatkan sebuah GroupBox yang didalamnya memiliki tiga buah RadioButton, dan masing – masing berfungsi untuk menampilkan tanggal dan hari (dalam bahasa Indonesia), jam digital (dengan memanfaatkan komponen timer) serta username dari sistem operasi Windows.

Gambar 9.18 Layout soal latihan StatusBar

Jawab :

1. **Buat form seperti pada gambar**
2. **Set property *interval* pada Timer menjadi 1000**
3. **Klik ganda pada control Timer dan ketikkan listing berikut pada prosedur *Timer1_Enabled***

```
StatusBar1.Panels(0).Text = Format(Now, "hh:mm:ss")
```

4. Klik ganda pada tombol *Refresh Status Bar* dan ketikkan listing berikut :

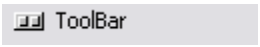
```
Dim xhari() As String = Split("Sabtu,Minggu," & _
   "Senin,Selasa,Rabu,Kamis,Jumat", ",")
   Timer1.Enabled = False
   Dim xtanggal As String = xhari(Weekday(Now. _
       Date)) & ", " & Format(Now.Date, "dd-MM-yyyy")
   For Each i As RadioButton In GroupBox1.Controls
       If i.Checked Then
           Select Case i.Text
               Case "Tampil tanggal dan hari"
                   StatusBar1.Panels(0).Text = xtanggal
               Case "Tampil jam digital"
                   Timer1.Enabled = True
               Case Else
                   StatusBar1.Panels(0).Text = _
                    "User name : " & _
                  System.Environment.UserName
           End Select
       End If
   Next
```

Namespace *System.Environment* digunakan untuk menampilkan informasi sistem operasi seperti versi Windows ataupun folder khusus seperti folder favourites dan lainnya.

Toolbar

Toolbar merupakan control yang berfungsi untuk menampilkan kumpulan icon atau tombol yang umumnya terletak di bagian atas sebuah form. Tombol yang terdapat dalam toolbar dapat diisi dengan icon, teks atau kombinasi dari keduanya.

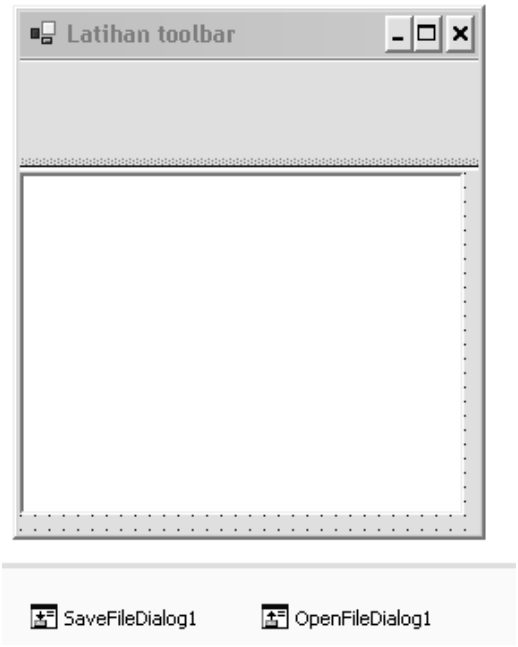
Toolbar sendiri seringkali dianalogikan dengan penggunaan menu, bahkan dalam banyak aplikasi, Toolbar lebih berfungsi sebagai sebuah shortcut dari menu yang sering digunakan dibandingkan berdiri sendiri sebagai sebuah kumpulan tombol.



Gambar 9.19 Toolbar dalam toolbox

Soal latihan :

Buat sebuah editor teks sederhana yang didalamnya terdapat sebuah Toolbar yang tombol-tombolnya dibuat secara dinamis dan berisi tombol untuk membuat file teks baru, membuka file teks, menyimpan file teks serta keluar dari aplikasi.



Gambar 9.20 Layout soal latihan ToolBar

Jawab :

1. **Buat form seperti pada gambar**
2. **Set property *MultiLine* pada Textbox menjadi *true* lalu perbesar ukurannya seperti pada gambar**
3. **Klik ganda pada form dan di prosedur *Form_Load* kemudian ketikkan listing berikut :**

```
Dim xteks() As String = _
    Split("New,Open,Save,Close", ",")
For i As Integer = 0 To 3
    Dim xbutton As New ToolBarButton
    ToolBar1.Buttons.Add(xbutton)
    xbutton.Text = xteks(i)
Next
```

Cara lain untuk menambahkan tombol dalam Toolbar adalah dengan mengisi property *Buttons*, Selain bisa diisi teks juga bisa diisi icon dengan memanfaatkan control *ImageList*.

4. **Klik ganda pada Toolbar dan ketikkan listing berikut ini :**

```
Dim xfilter As String = "Teks (*.txt)|*.txt"
Select Case ToolBar1.Buttons.IndexOf(e.Button)
    Case 0
        TextBox1.Text = ""
    Case 1
        Dim xteks As System.IO.StreamReader
        With OpenFileDialog1
            .Filter = xfilter
            .ShowDialog()
            xteks = System.IO.File.OpenText(.FileName)
        End With
        TextBox1.Text = xteks.ReadToEnd
        xteks.Close()
    Case 2
        Dim xteks As System.IO.StreamWriter
        With SaveFileDialog1
            .Filter = xfilter
            .ShowDialog()
            xteks = System.IO.File.CreateText(.FileName)
        End With
        xteks.Write(TextBox1.Text)
        xteks.Close()
    Case 3
        Close()
End Select
```

þ Keterangan

Manipulasi file teks menggunakan namespace System.IO.Untuk membaca teks, digunakan object StreamReader, sedangkan untuk menuliskan file digunakan object StreamWriter.Sedangkan untuk penyimpanan digunakan namespace System.IO.File.
Hal terpenting yang perlu diingat adalah bahwa tiap penggunaan namespace System.IO harus diakhiri dengan perintah close untuk object yang telah dideklarasikan sebelumnya.
Komponen SaveFileDialog digunakan untuk membuka kotak dialog serta menyimpan file dengan format yang telah ditentukan sebelumnya.

Studi Kasus Pengembangan

Tujuan :

- **Mampu mengimplementasikan dasar teori ke dalam tugas studi kasus pengembangan**
- **Mampu mendemonstrasikan kemampuan rekayasa perangkat lunak ke dalam sebuah aplikasi secara efektif dan efisien**

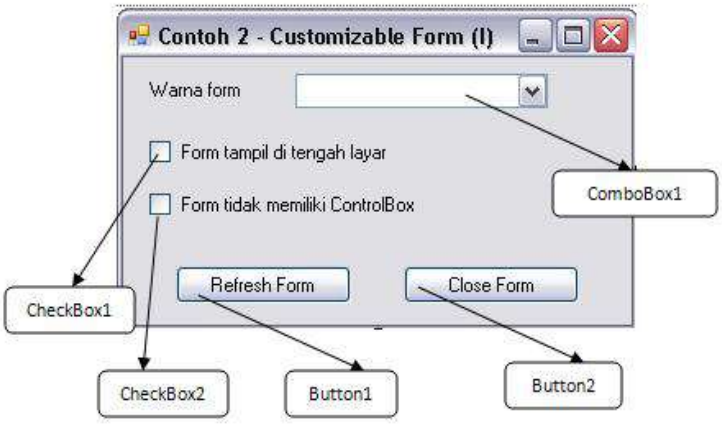
Kasus 1 : Customizable Form

Level : Pemula

Tujuan :

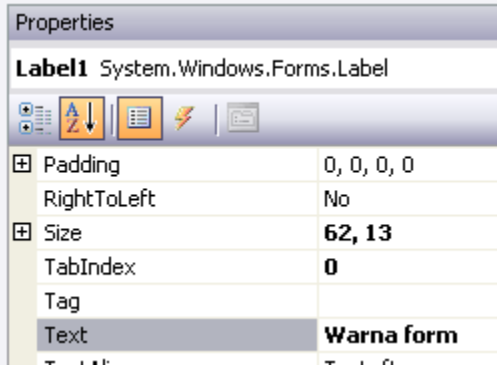
- **Membuat form yang dapat diatur tampilannya melalui setting property runtime**
- **Mengenalkan property form**
- **Mengenalkan komponen Checkbox dan logika percabangan**
- **Mengenalkan komponen Combobox**

Desain form awal :

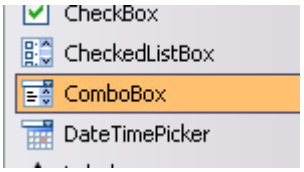


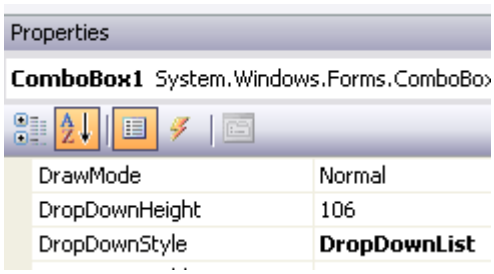
Langkah penyelesaian :

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.**

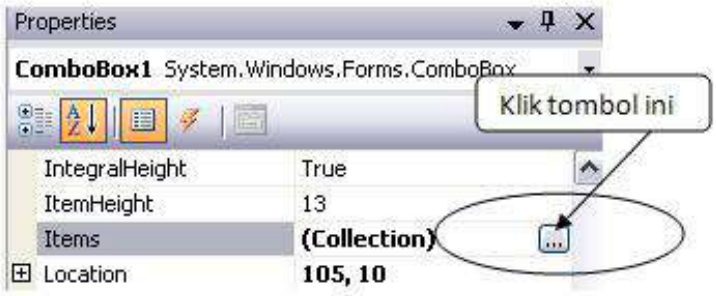




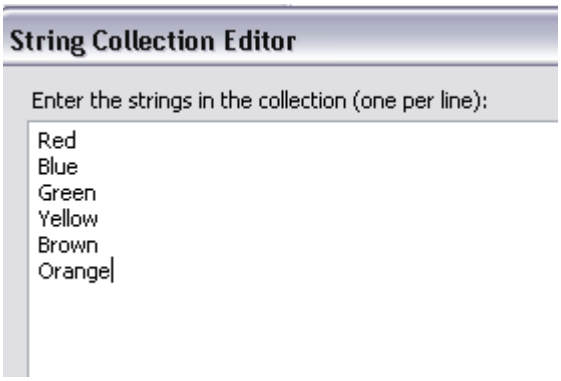
þ Keterangan

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.

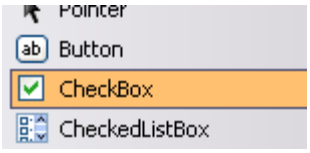
3. **Selanjutnya klik button kecil di dalam property Items untuk menambahkan item di dalam combobox tersebut.**

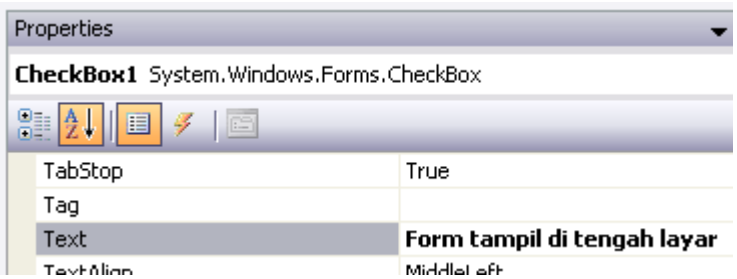


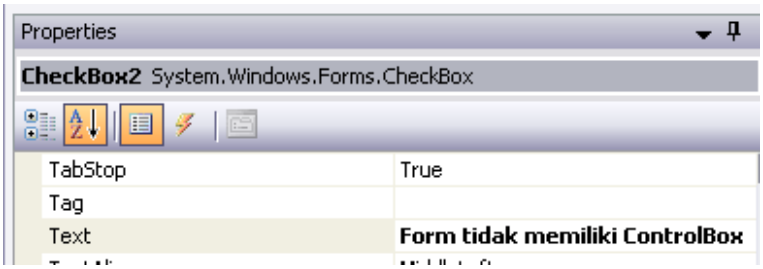
4. **Lalu isikan item berikut ke dalam item combobox.**



5. **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.**



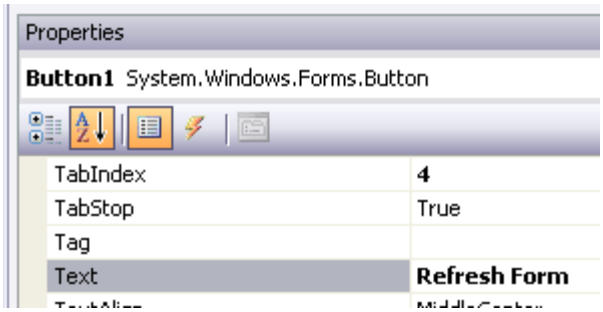


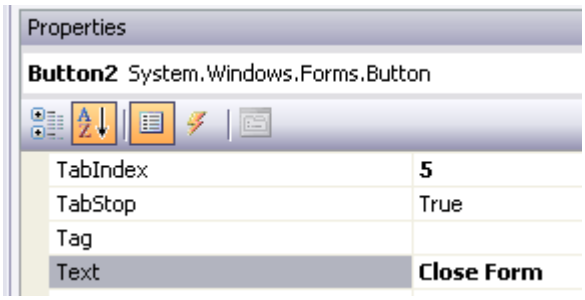


þ Keterangan

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.

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





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

```
ComboBox1.SelectedIndex = 0
```

þ Keterangan

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.

8. **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()
```

9. **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
```

þ Keterangan

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).

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

11. **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.**

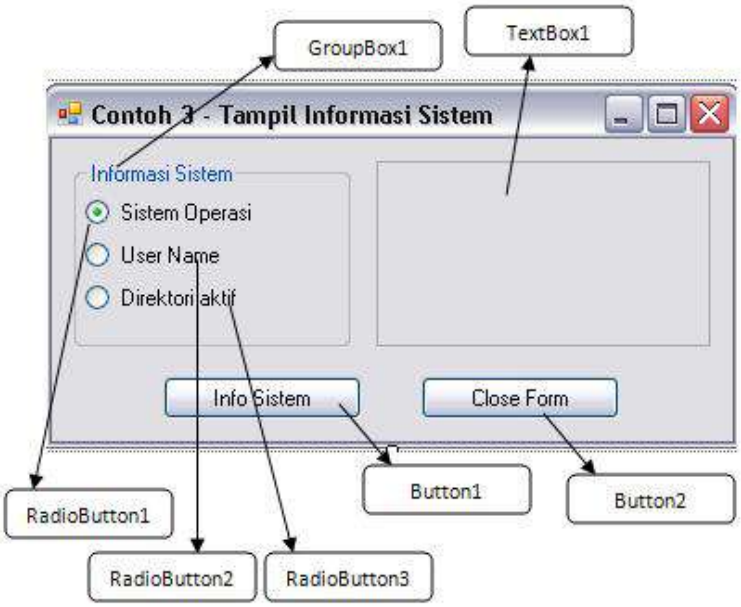
Kasus 2 : Menampilkan Informasi Sistem

Level : Pemula

Tujuan :

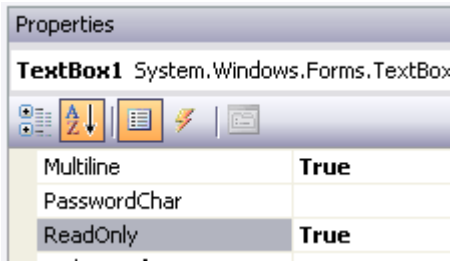
- **Membuat form yang dapat diatur tampilannya.**
- **Mengenalkan komponen Radiobutton dan GroupBox serta penggunaan logika perulangan For Each untuk melakukan pengecekan dalam sebuah container**
- **Mengenalkan penggunaan kata kunci My untuk mendapatkan informasi sistem**
- **Mengenalkan logika percabangan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

Desain form awal :



Langkah penyelesaian :

1. **Seperti halnya contoh sebelumnya, buat sebuah solution baru dengan nama *Contoh2*.**
2. **Pada form yang tersedia, drag komponen-komponen berikut ini :**
   a. **Sebuah textbox, lalu set property *ReadOnly* menjadi *True* seperti pada gambar berikut :**

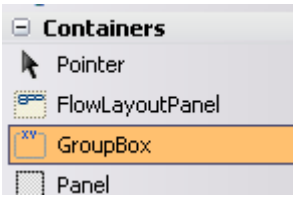


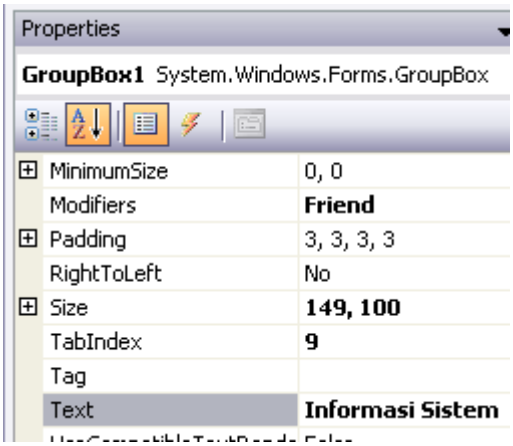
þ Keterangan

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*.**



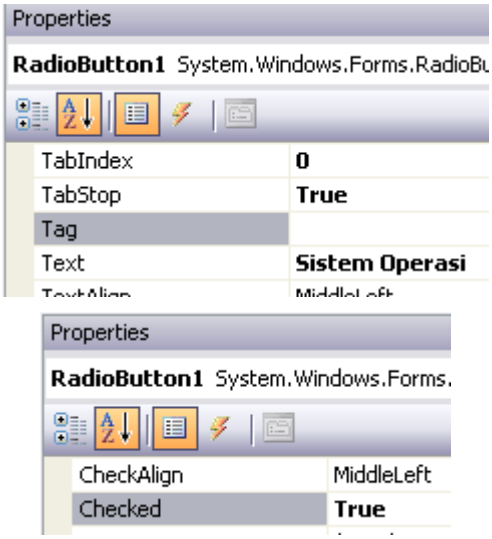


þ Keterangan

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**

þ Keterangan

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
```

þ Keterangan

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 :


Contoh 3 - Tampil Informasi Sistem
Informasi Sistem
Sistem Operasi
User Name
Direktori aktif
Sistem Operasi :
Microsoft Windows XP Professional
Info Sistem
Close Form

Kasus 3 : Random Star Screensaver

Level : Menengah

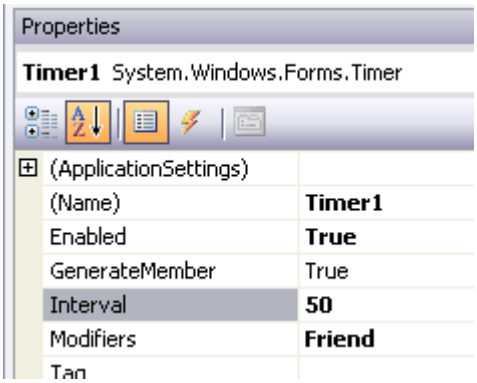
Tujuan :

- **Membuat aplikasi screensaver berupa karakter bintang (*) yang secara acak (baik posisi maupun warna) muncul di seluruh layar.**
- **Mengenal penggunaan bilangan acak**
- **Mengenal setting posisi dan warna dari sebuah komponen secara runtime**
- **Mengenal penggunaan Timer**
- **Mengenal pendeteksian kursor mouse**
- **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.**



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

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

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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 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. Untuk**

melakukan pengecekan, berpindahlah ke desktop Windows, dan klik kanan lalu pilih sub menu Properties. 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 !!

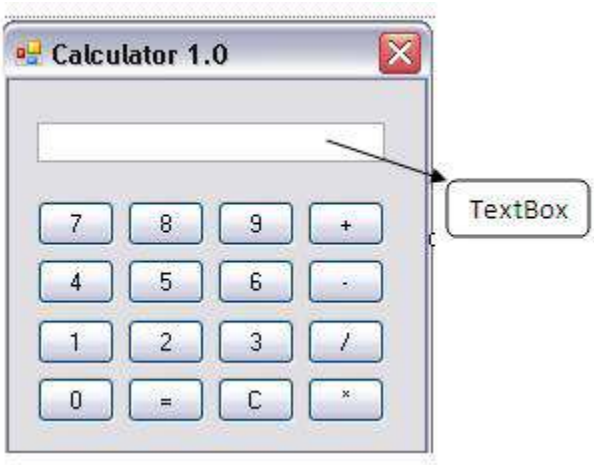
Kasus 4 : Kalkulator Sederhana

Level : Menengah

Tujuan :

- **Membuat aplikasi tiruan kalkulator dalam mode yang lebih sederhana**
- **Mengaplikasikan penggunaan prosedur dan fungsi dalam sebuah aplikasi**
- **Mengaplikasikan penggunaan prosedur dalam komponen secara dinamis**
- **Menggunakan percabangan majemuk**
- **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.**

   þ Keterangan

   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
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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.

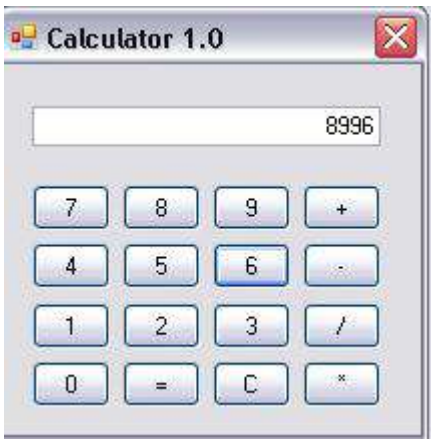
6. **Prosedur terpenting di dalam aplikasi kalkulator sederhana ini adalah prosedur untuk melakukan perhitungan berdasarkan operasi matematika yang dipilih oleh pengguna. Sebelum percabangan dieksekusi, maka dilakukan terlebih dulu pengecekan terhadap angka-angka yang telah dimasukkan. Karena dalam kalkulator sederhana ini hanya memperbolehkan operasi matematika yang membutuhkan dua angka (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.**



Kasus 5 : Editor Sederhana

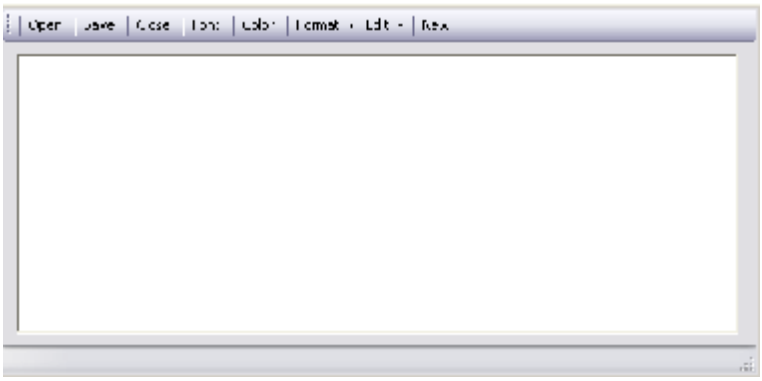
Level : Menengah

Tujuan :

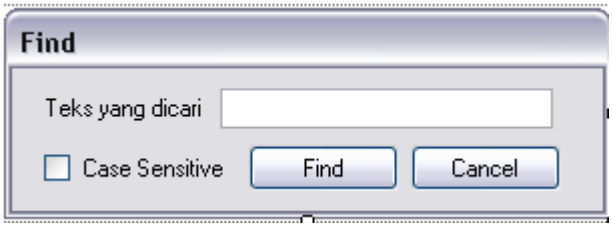
- **Membuat aplikasi editor rich text format sederhana**
- **Mengaplikasikan teknik komponen rich text box**
- **Mengaplikasikan teknik pemakaian kotak dialog dalam aplikasi (open, save,color dan font)**
- **Menggunakan komponen timer status strip dan tool strip**
- **Mengenal manipulasi string dalam komponen rich text box**
- **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**

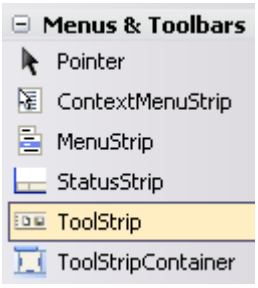


2. **Desain frmFind**



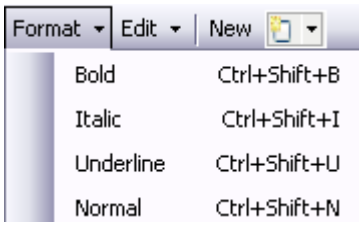
Langkah penyelesaian untuk form utama :

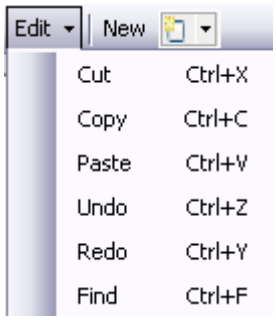
1. **Pada form pertama (frmUtama), drag sebuah komponen *ToolStrip* ke dalam form dan isikan didalamnya item dengan struktur sebagai berikut :**



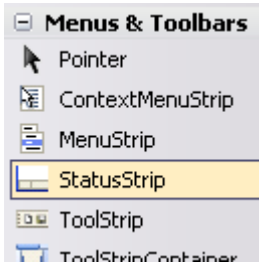
þ Keterangan

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

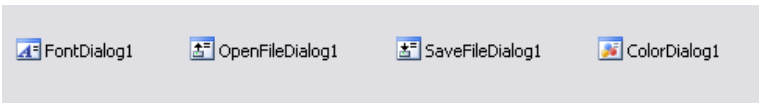




2. **Selanjutnya drag sebuah komponen *StatusStrip* ke dalam form.**



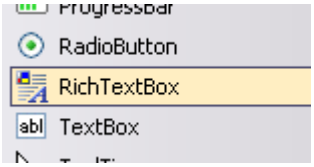
3. **Untuk kepentingan penyimpanan dan pengeditan teks drag masing-masing komponen untuk kotak dialog yaitu komponen : *FontDialog*, *ColorDialog*. *OpenFileDialog* dan *SaveDialog*.**



þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

Dalam deklarasi awal variabel dengan jangkauan di dalam satu form tersebut, terdapat empat buah variabel yaitu :
JudulAplikasi yang digunakan untuk menampilkan caption pada form
Ekstension yang berfungsi untuk menentukan ekstensi file default pada saat kotak dialog open/save file teks
filterFile yang digunakan untuk filter otomatis pada saat kotak dialog open/save file teks
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
```

þ Keterangan

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
```

þ Keterangan

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) :

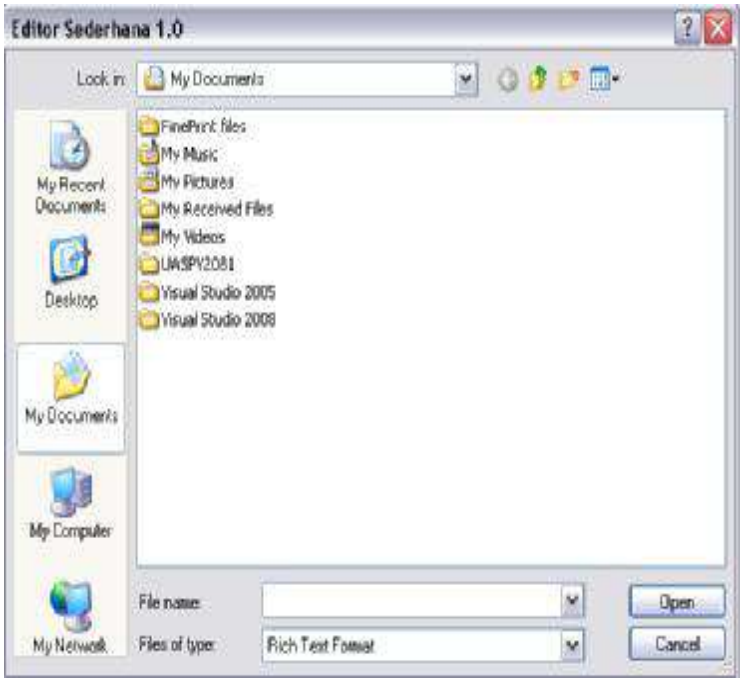
Item Open

```
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
```

þ Keterangan

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()
```

þ Keterangan

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
```

þ Keterangan

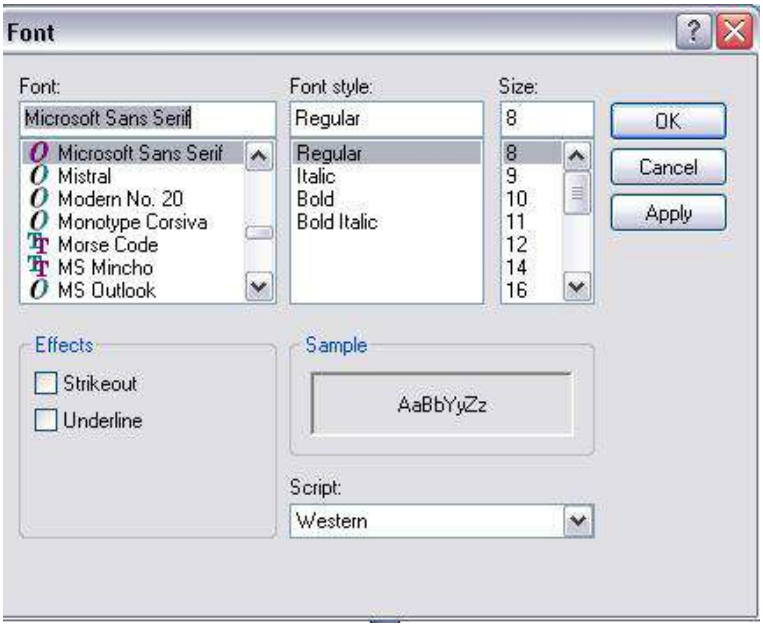
Sesungguhnya apa yang dilakukan pada saat aplikasi ditutup adalah menampilkan konfirmasi jika ternyata file teks yang sedang diedit di dalam RichTextBox belum disimpan oleh pengguna. Jika memang teks tersebut belum disimpan, maka pengguna akan dipaksa untuk menjawab kotak konfirmasi untuk melakukan penyimpanan 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
```

þ Keterangan

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 :

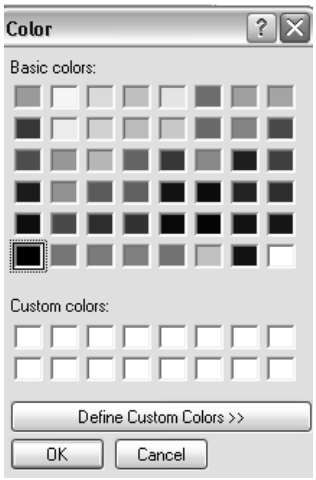


Item Color

```
With ColorDialog1
    .Color = RichTextBox1.SelectionColor
    If .ShowDialog() = _
        Windows.Forms.DialogResult.OK Then
        RichTextBox1.SelectionColor = .Color
    End If
End With
```

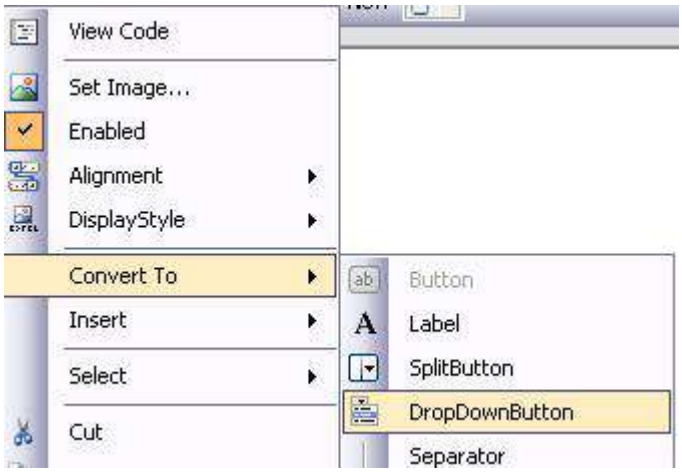
þ Keterangan

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 :



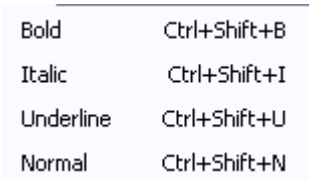
Item Format

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*.**



þ Keterangan

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
```

þ Keterangan

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)
```

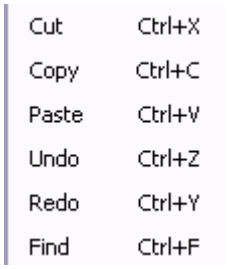
```
        Case "Normal"
            .SelectionFont = _
            New Drawing.Font _
            (.SelectionFont, FontStyle.Regular)
    End Select
End With
```

þ Keterangan

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 :



þ Keterangan

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
```

þ Keterangan

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 = ""
```

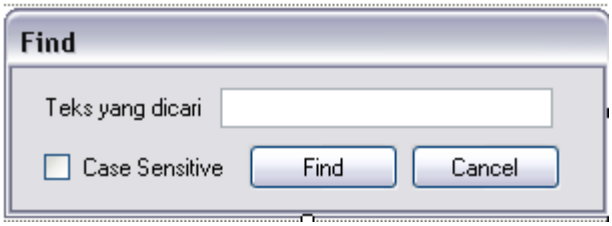
```
refreshStatus()
```

þ Keterangan

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()
```

þ Keterangan

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.

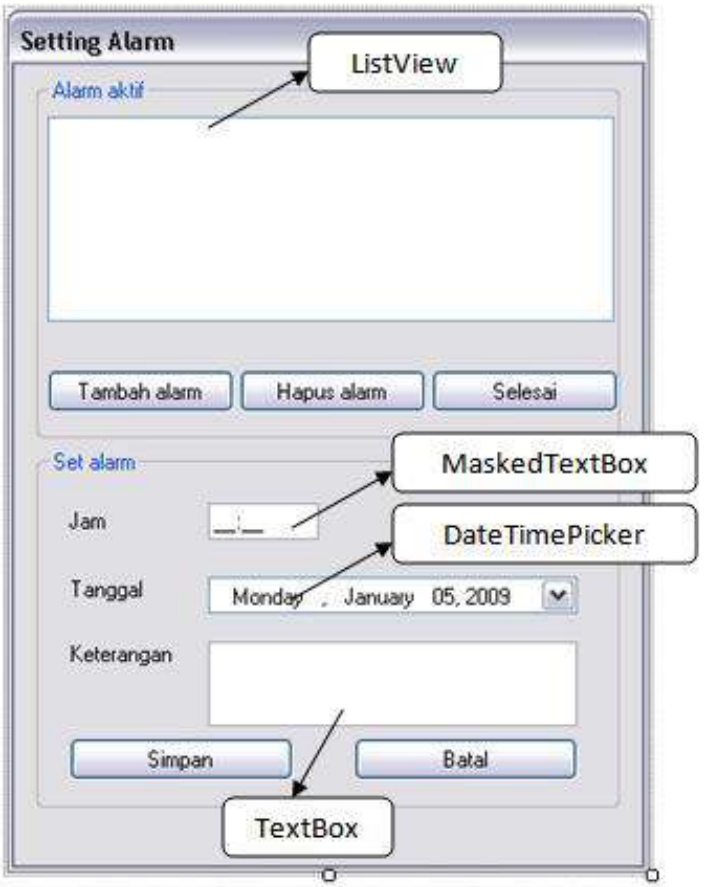
Kasus 6 : Alarm

Level : Menengah

Tujuan :

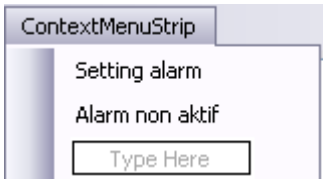
- **Membuat aplikasi alarm notification yang mampu melakukan alert atau peringatan pada waktu tertentu yang datanya telah disimpan dalam sebuah file teks**
- **Menggunakan komponen NotifyIcon dalam sebuah aplikasi model stay residence**
- **Mengenal manipulasi file teks secara sekuensial**
- **Mengenal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

Desain awal :



Langkah penyelesaian :

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



þ Keterangan

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
```

þ Keterangan

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()
```

þ Keterangan

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**

þ Keterangan

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**

þ Keterangan

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()
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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)
```

þ Keterangan

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
```

þ Keterangan

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.

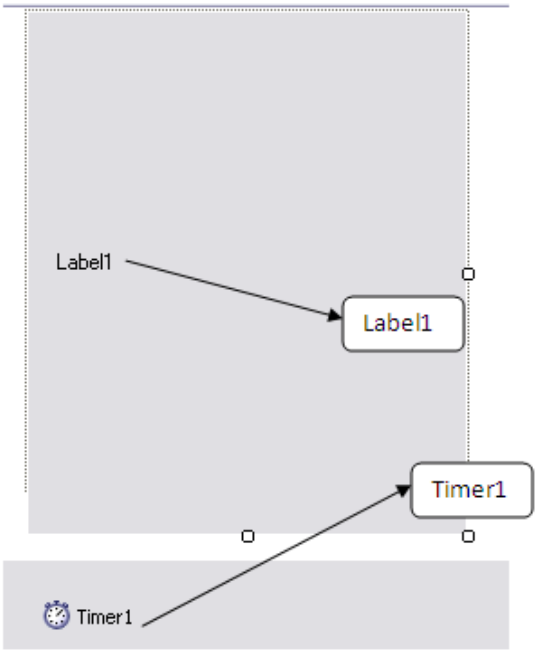
Kasus 7 : Marquee Screensaver

Level : Menengah

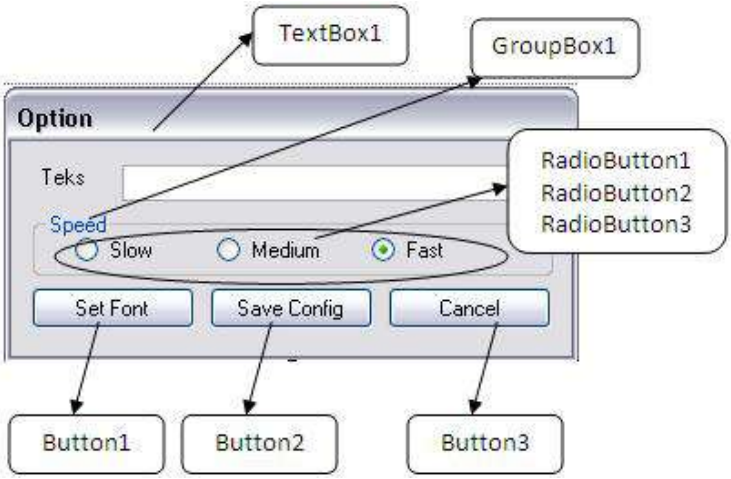
Tujuan :

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

þ Keterangan

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**

þ Keterangan

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 :**

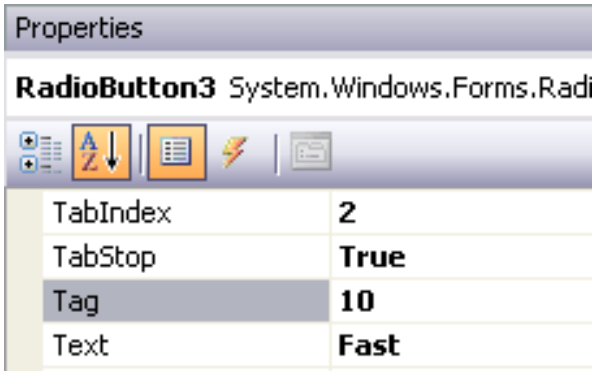
      i. **Radiobutton *Slow* à 50**

      ii. **Radiobutton *Medium* à 25**

      iii. **Radiobutton *Fast* à 10**

þ Keterangan

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
```

þ Keterangan

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"
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

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.**

þ Keterangan

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
```

þ Keterangan

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
```

þ Keterangan

Pada button atau tombol *Set Font* saat diklik akan menampilkan kotan 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()
```

þ Keterangan

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.

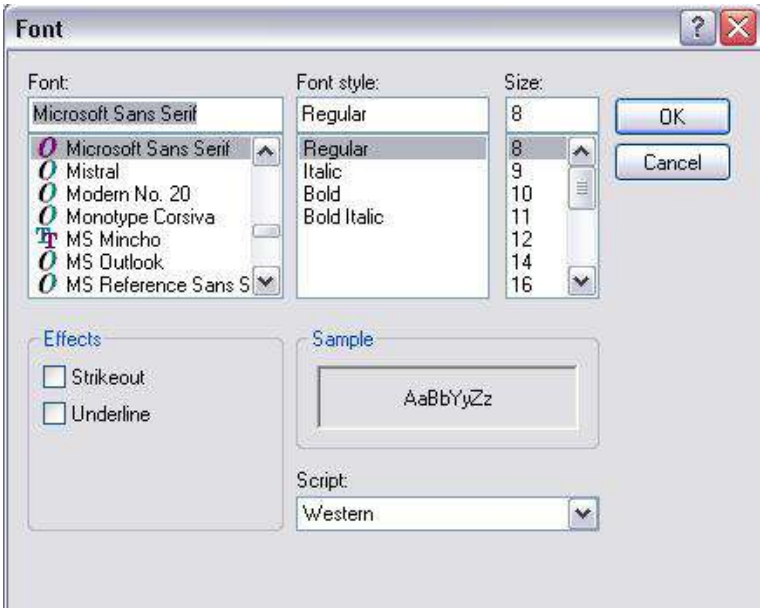
Dan untuk konfigurasi yang terakhir adalah konfigurasi tulisan yang akan ditampilkan oleh screensaver. Secara default, jika tulisan tidak diisi oleh pengguna, maka tulisan di dalam screensaver akan diisi dengan tulisan *Learning By Sample*.

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.**

Kasus Tugas Mandiri

Tujuan :

- **Menyajikan kasus-kasus pemrograman yang tidak ditampilkan solusinya untuk diselesaikan sebagai tugas mandiri**
- **Kasus yang ditampilkan merupakan kasus dari level menengah, bukan lagi dari level pemula**

Word Cube

Word cube merupakan kubus dari sebuah kata palindrom (jika dibalik akan memiliki ejaan yang sama) yang ditampilkan dalam sebuah komponen Textbox dengan mode MultiLine. Kata yang ditampilkan berasal dari inputan dalam sebuah Textbox yang berbeda, dan akan dipicu sebuah tombol untuk membentuk kubus kata tersebut.

Sebagai contoh, jika diinputkan kata KATAK, maka hasilnya adalah sebagai berikut :

```
K A T A K
A       A
T       T
A       A
K A T A K
```

Proses yang harus dilakukan adalah :

5. **Melakukan pengecekan apakah kata yang diinputkan adalah palindrom**
6. **Menempatkan kata dalam Textbox melalui proses looping dengan petunjuk sebagai berikut :**
    a. **Pada baris pertama letakkan seluruh kata**
    b. **Pada baris berikutnya, ambil huruf kedua dan beri spasi sebanyak panjang kata lalu letakkan ulang huruf kedua tersebut.**
    c. **Ulangi langkah 2b hingga sejumlah huruf dalam kata dikurangi dengan huruf awal dan akhir**
    d. **Pada baris terakhir letakkan lagi seluruh kata.**

Simple Encryption

Jenis algoritma untuk enkripsi sangatlah beragam, dalam studi kasus ini diaplikasikan modifikasi dari algoritma shift transposition. Dari inputan sebuah kalimat, maka akan dienkripsi dengan menggunakan tabel subtitusi yang kemudian "digeser" sebanyak satu kolom dari tabel yang yang sudah disediakan. Dan juga diharuskan agar aplikasi yang dibuat dapat melakukan dekripsi dari kalimat yang sudah dienkrip tersebut.

Tabel subtitusi adalah sebagai berikut :

Huruf asli	**:**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Subtitusi	**:**	**T**	**S**	**R**	**Q**	**P**	**O**	**N**	**M**	**L**	**K**
Huruf asli	**:**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**
Subtitusi	**:**	**‘**	**“**	**,**	**.**	**Z**	**Y**	**X**	**W**	**V**	**U**
Huruf asli	**:**	**U**	**V**	**W**	**X**	**Y**	**Z**	**.**	**,**	**“**	**‘**
Subtitusi	**:**	**J**	**I**	**H**	**G**	**F**	**E**	**D**	**C**	**B**	**A**

Sebagai contoh, jika diinputkan kalimat STIKOM SURABAYA, maka akan menjadi sebagai berikut :

S disubtitusi menjadi V dan digeser satu kali ke kanan menjadi U

T disubtitusi menjadi U dan digeser dua kali ke kanan menjadi “

I disubtitusi menjadi L dan digeser tiga kali ke kanan menjadi S

K disubtitusi menjadi ‘ dan digeser empat kali ke kanan menjadi Z

O disubtitusi menjadi Z dan digeser lima kali ke kanan menjadi U

M disubtitusi menjadi , dan digeser enam kali ke kanan menjadi V

S disubtitusi menjadi V dan digeser tujuh kali ke kanan menjadi Y

U disubtitusi menjadi J dan digeser delapan kali ke kanan menjadi B

R disubtitusi menjadi W dan digeser sembilan kali ke kanan menjadi X

A disubtitusi menjadi T dan digeser sepuluh kali ke kanan menjadi T

B disubtitusi menjadi S dan digeser sebelas kali ke kanan menjadi R

A disubtitusi menjadi T dan digeser duabelas kali ke kanan menjadi R

Y disubtitusi menjadi F dan digeser tigabelas kali ke kanan menjadi C

A disubtitusi menjadi T dan digeser empatbelas kali ke kanan menjadi P

Sehingga hasil dari enkripsi adalah :

STIKOM SURABAYA à U”SZUV YBXTRRCP

Logika Perhitungan (I)

Diberikan kondisi untuk sebuah form yang akan melakukan perhitungan harga jual bersih dan jumlah barang yang akan dikirim ke pelanggan. Kondisi yang harus dipenuhi adalah :

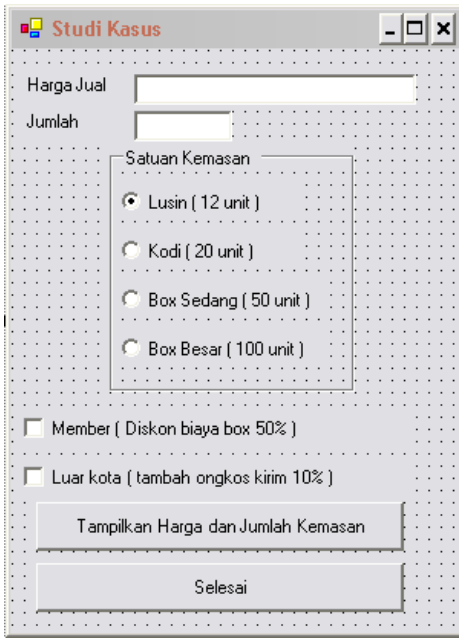
1. **Tiap pengiriman ditentukan satuan kemasannya berdasarkan jumlah barang. Jika barang yang dikemas ternyata memiliki sisa, maka akan dikemas ke kemasan yang lebih kecil. Misal : jika memilih kemasan box sedang dan ternyata masih ada sisa barang sebanyak 15, maka akan dimasukkan ke dalam kemasan kodi, tetapi jika sisa barang hanya sebanyak 10, maka dimasukkan ke dalam kemasan lusin.**
2. **Setiap member mendapat diskon biaya tiap box sebesar 50%**
3. **Setiap pengiriman ke luar kota mendapat tambahan biaya ongkos kirim sebesar 10% dari total ongkos barang yang dikirim**
4. **Harga bersih dan jumlah kemasan dan barang sisa ditampilkan dalam sebuah message box.**

Untuk harga tiap box adalah sebagai berikut :

5. **Lusin : Rp. 1.000**
6. **Kodi : Rp. 2.000**
7. **Box sedang : Rp. 5.000**
8. **Box besar Rp. 7.500**

Contoh : jika dibeli 150 unit barang dan serta lokasi pembeli terletak di luar kota, maka pembeli yang tidak menjadi member akan mendapat harga pengiriman bersih sebesar Rp. 16.500 dengan rincian kemasan kodi sebanyak 7 kemasan seharga Rp. 14.000 serta sisa barang sebanyak 10 barang yang harus dimasukkan ke dalam kemasan lusin sebanyak 1 seharga Rp. 1.000.

Layout dari form yang akan dikerjakan adalah sebagai berikut :

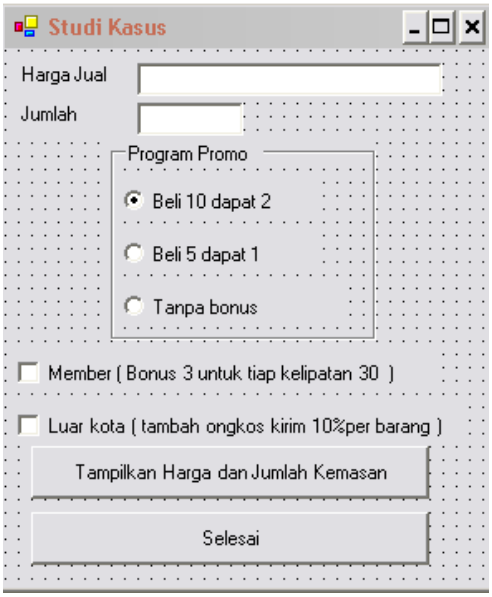

Studi Kasus
Harga Jual
Jumlah
Satuan Kemasan
Lusin (12 unit)
Kodi (20 unit)
Box Sedang (50 unit)
Box Besar (100 unit)
Member (Diskon biaya box 50%)
Luar kota (tambah ongkos kirim 10%)
Tampilkan Harga dan Jumlah Kemasan
Selesai

Diberikan kondisi untuk sebuah form yang akan melakukan perhitungan harga jual bersih dan jumlah barang yang akan dikirim ke pelanggan. Kondisi yang harus dipenuhi adalah :

1. **Untuk program promo terdapat dua pilihan : *Beli 10 dapat 2* atau *Beli 5 dapat 1***
2. **Setiap member mendapat bonus 3 unit dari setiap 30 unit barang yang dibeli**
3. **Setiap pengiriman ke luar kota mendapat tambahan biaya ongkos kirim sebesar 10% dari tiap barang yang didapat**
4. **Harga bersih dan jumlah barang ditampilkan dalam sebuah message box.**

Contoh : jika dibeli 50 unit barang dengan harga satuan sebesar Rp. 10.000 dan mendapat program promo *Beli 5 dapat 1* serta lokasi pembeli terletak di luar kota, maka pembeli yang menjadi member akan mendapat harga bersih sebesar Rp. 562.000 dan jumlah barang sebanyak 62.

Layout dari form yang akan dikerjakan adalah sebagai berikut :



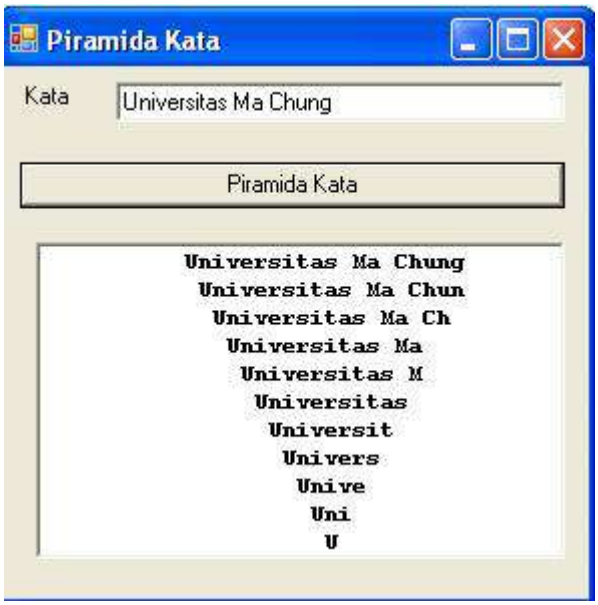
Piramida Kata

Latihan berikut ini mirip dengan latihan soal untuk piramida bilangan pada bab mengenai listbox, perbedaan utama dari latihan soal yang pertama adalah variabel yang akan dijadikan piramida bukan lagi deret bilangan melainkan dari sebuah kata atau lebih.

Dari sebuah kata atau lebih, yang diinputkan ke dalam sebuah textbox, maka berikutnya akan dibuat sebuah piramida berdasarkan panjang kata serta pemotongan huruf yang ada dalam rangkaian kata tersebut. Sedangkan piramida yang akan ditampilkan memiliki susunan terbalik, sehingga akan tampak seperti sebuah segitiga dengan ujung yang menghadap ke bawah.

Layout awal dari form adalah sebagai berikut :

Sedangkan hasil yang nantinya akan tampil adalah sebagai berikut :


Piramida Kata
Kata
Universitas Ma Chung
Piramida Kata
Universitas Ma Chung
Universitas Ma Chun
Universitas Ma Ch
Universitas Ma
Universitas M
Universitas
Universit
Univers
Unive
Uni
U

Expression Evaluator

Diinginkan sebuah aplikasi untuk melakukan evaluasi terhadap sebuah ekspresi aritmetika dengan memperhatikan derajat operasi yang ada, yaitu : perkalian, pembagian, penjumlahan dan pengurangan. Serta memperhatikan pula operasi dalam tanda kurung yang berarti harus diselesaikan terlebih dulu sebelum operasi yang lain. Ekspresi atau formula diketikkan secara utuh dalam sebuah Textbox dan hasilnya ditampilkan dalam sebuah kotak pesan.

Sebagai contoh, jika diinputkan ekspresi berikut :

(20+5)-1*2+45
akan menghasilkan 68 dengan rincian langkah sebagai berikut :

- 20 + 5 = 25 (karena ada dalam parentheses atau tanda kurung)
- 1 * 2 = 2 (karena derajat perkalian lebih tinggi dari penjumlahan dan pengurangan)
- 25 – 2 = 23 (hasil dari operasi dalam tanda kurung digabung dengan hasil perkalian)
- 23 + 45 = 68

Daftar Pustaka

Balena, Fransesco, (2006), *Programming Microsoft Visual Basic 2005,* Microsoft Press

Deitel, Paul J and Harvey J. Deitel, (2006), *Visual Basic 2005 for Programmers second edition* ,Prentice Hall

Fouche, Guy and Trey Nash, (2007), *Accelerated VB 2005*, Apress

Herman, Todd et al, (2007), *Visual Basic 2005 Recipes*, Apress

Jones, A. Russel and Mike Gunderloy, (2004), *.NET Programming 10 Minutes Solutions*, Sybex

Lee, Wei Meng, (2005), *Visual Basic 2005 Jumpstart*, O'Reilly

Lomax, Paul et al, (2006), *Visual Basic 2005 in a Nutshell:3rd edition*, O'Reilly

MacDonald, Matthew, (2006), *The Book of Visual Basic 2005*, No Starch Press

Patrick, Tim, (2006), *Start to Finish Visual Basic 2005 : Learn Visual Basic 2005 as you design and develop complete application*, Addison Wesley

Pelland, Patrice, (2006), *Build a Program Now : Microsoft Visual Basic 2005 Express Edition*, Microsoft Press

Sells, Chris and Michael Weinhardt, (2006), *Windows Forms 2.0 Programming*, Addison Wesley

Glosarium

Istilah	*Definisi*
.NET Framework	**Kerangka yang mendukung pengembangan perangkat lunak serta menjadi bagian yang terintegrasi dalam sistem operasi Windows.**
CLR (Common Language Runtime)	**Layer pertama yang terdapat di dalam .NET Framework dan bertanggungjawab atas pengaturan memory, *garbage collection, structured exception handling* dan *multithreading*.**
CLS (Common Language Specification)	**Fitur minimum yang harus dimiliki oleh bahasa pemrograman untuk dapat berjalan di atas .NET Framework**
CTS (Common Type System)	**Semua tipe data inti dan mekanisme inti yang digunakan dalam sebuah aplikasi yang menggunakan .NET**
.NET Class Library	**Class-class yang disediakan oleh .NET yang berisi fungsi-fungsi mulai dari yang umum hingga kompleks**
Windows Form	**Bagian dari .NET Framework yang memperbolehkan programmer untuk membuat aplikasi Win32, baik yang berjalan secara *stand alone* ataupun *client-server*.**
GDI+	**Bagian dari .NET Framework yang berhubungan dengan pengolahan citra dan komputer grafika**
Solution	**Hirarki tertinggi dalam pembuatan aplikasi di Visual Basic .NET**
Project	**Bagian utama dari sebuah aplikasi dalam Visual Basic .NET**
Property	**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.**

Istilah	*Definisi*
Methods	**Prosedur yang diasosiasikan ke sebuah object atau seringkali ke sebuah komponen.**
Event	**Hasil dari sebuah tindakan oleh pengguna terhadap suatu komponen.**
Form	**Interface utama dari sebuah aplikasi windows standar di dalam Visual Basic .NET**
Variabel	**Penampung sementara dari sebuah tipe data tertentu di memori komputer**
Scope	**Masa sebuah variabel tersebut bisa digunakan dalam sebuah solution di Visual Basic .NET.**
Array	**Set variabel yang dapat diakses dengan mengidentifikasi masing – masing item array dengan indeks angka tertentu.**
Prosedur	**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.**
Fungsi	**Prosedur yang mengembalikan nilai balik saat pemanggilan berlangsung**
Handles	**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).**
SDI (Single Document Interface)	**Mengasumsikan bahwa tiap form akan berdiri sendiri tanpa struktur induk – anak (parent – child), sehingga programmer lebih bebas menentukan form mana yang akan diaktifkan.**
MDI (Multiple Document Interface)	**Menyebabkan semua form yang dianggap sebagai child akan selalu berada di dalam form induk atau form parent.**
Field Validation	**Mengasumsikan penanganan kesalahan dilakukan di tiap aksi pengguna pada masing – masing control.**

Istilah	*Definisi*
Form Validation	**Melakukan pengecekan kesalahan di akhir proses dari sebuah form. Umumnya pengecekan dilakukan di tombol terakhir yang menyatakan proses seperti tombol untuk simpan.**

Indeks